CONVERSION OF SIGN LANGUAGE TO TEXT AND SPEECH USING DEEP LEARNING

PROJECT REPORT

Submitted by

NELSON NICHOLAS M	**REG NO: 510420104062**
RAFFI A	**REG NO: 510420104075**
SARMA S	**REG NO: 510420104089**
MADAN RAJ B	**REG NO: 510420104057**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ARUNAI ENGINEERING COLLEGE

TIRUVANNAMALAI- 606 603

ANNA UNIVERSITY: CHENNAI 600025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled "**CONVERSION OF SIGN LANGUAGE TO TEXT AND SPEECH USING DEEP LEARNING**" is the bonafide work of **NELSON NICHOLAS M (510420104062) , RAFFI A (510420104075) , SARMA S (510420104089) , MADAN RAJ B (510420104057)** who carried out the project work under our supervision, for the partial fulfilment of the requirements for the AWARD OF THE DEGREE OF **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING.** Certified further, that to the best of our knowledge and belief, work reported herein does not form part of any other thesis or dissertation on the basis which a degree or an award was conferred on an earlier occasion.

SIGNATURE OF THE SUPERVISOR

Mrs. G.SHOBA, M.E.,
Assistant Professor,
Computer Science and Engineering,
Arunai Engineering College,
Tiruvannamalai- 606 603.

SIGNATURE OF HOD

Mrs. V.UMADEVI, M.E.,
Head of the Department,
Computer Science and Engineering,
Arunai Engineering College,
Tiruvannamalai- 606 603.

INTERNAL EXAMINER

EXTERNAL EXAMINER

CERTIFICATE OF EVALUATION

COLLEGE NAME : 5104 - ARUNAI ENGINEERING COLLEGE TIRUVANNAMALAI

BRANCH : B.E , COMPUTER SCIENCE AND ENGINEERING

SEMESTER : 6TH SEM

SUB CODE & NAME : CS8611 - MINI PROJECT

DATE OF EXAM :

NAME OF THE STUDENTS	REG.NO.	TITLE OF THE PROJECT	NAME OF THE INTERNAL GUIDE
NELSON NICHOLAS M RAFFI A SARMA S MADAN RAJ B	510420104062 510420104075 510420104089 510420104057	CONVERSION OF SIGN LANGUAGE TO TEXT AND SPEECH USING DEEP LEARNING	Mrs. G.SHOBA Asst Prof / CSE

The report of the **Project Work** submitted for the fulfilment of Bachelor of Engineering degree in **COMPUTER SCIENCE AND ENGINEERING** of Anna university was evaluated and confirmed to be report of the work done by the above student.

Submitted for university examination held on _____

SIGNATURE OF THE SUPERVISOR

Mrs. G.SHOBA, M.E.,
Assistant Professor,
Computer Science and Engineering,
Arunai Engineering College,
Tiruvannamalai- 606 603.

SIGNATURE OF HOD

Mrs. V.UMADEVI, M.E.,
Head of the Department,
Computer Science and Engineering,
Arunai Engineering College,
Tiruvannamalai- 606 603.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT:

It is my duty to thank the GOD Almighty and my beloved parents and teachers for their persistent blessing and constant encouragement to reach this level of what I am today.

A project of this nature needs co-operation and support from many for successful completion. In this regard, I am fortunate to express my heartfelt thanks to our beloved Founder Chairman **Mr.E.V.VELU,** Chair Person **Mrs.SANKARI VELU** and **Er.E.V.KUMARAN M.E.,** Vice Chairman, for providing necessary facilities with highclass environment throughout the course.

I take the privilege of expressing my sincere thanks to our beloved Principal, **Dr.R.RAVICHANDARAN M.E.,Ph.D** for granting permission to undertake the project.I am fortunate to express my heartfelt thanks to our beloved Registrar **Dr.R.SATHIYASEELAN M.E.,Ph.D** for providing necessary facilitiesin accomplishing this project work.

I am most grateful to **Mrs. V.UMADEVI, M.E.,** Head of the Department**,** Department of Computer Science and Engineering, who has given me both moral and technical support adding experience to the job I have undertaken.

I thank my project supervisor **Mrs. G.SHOBA, M.E.,** Asst Prof/CSE for her astonishing guidance and encouragement for the successful completion of this project. They held me to the highest standards of quality and accuracy.

Last but not the least I would like to thank my family members, friends, teaching and non-teaching staffs, who has assisted us directly or indirectly throughout this project work.

ABSTRACT

Sign language is one of the oldest and most natural form of language for communication, hence we have come up with a real time method using neural networks for finger spelling based American sign language. Automatic human gesture recognition from camera images is an interesting topic for developing vision. We propose a convolution neural network (CNN) method to recognize hand gestures of human actions from an image captured by camera. The purpose is to recognize hand gestures of human task activities from a camera image. The position of hand and orientation are applied to obtain the training and testing data for the CNN. The hand is first passed through a filter and after the filter is applied where the hand is passed through a classifier which predicts the class of the hand gestures. Then the calibrated images are used to train CNN.

சுருக்கம்

சைகை மொழி என்பது தகவல்தொடர்புக்கான மொழியின் பழமையான மற்றும் இயற்கையான வடிவங்களில் ஒன்றாகும், எனவே விரல் எழுத்துப்பிழை அடிப்படையிலான அமெரிக்க சைகை மொழிக்கான நரம்பியல் நெட்வொர்க்குகளைப் பயன்படுத்தி நிகழ்நேர முறையை நாங்கள் கொண்டு வந்துள்ளோம். கேமரா படங்களிலிருந்து தானியங்கி மனித சைகை அங்கீகாரம் பார்வையை வளர்ப்பதற்கான ஒரு சுவாரஸ்யமான தலைப்பு. கேமராவால் எடுக்கப்பட்ட படத்திலிருந்து மனித செயல்களின் கை சைகைகளை அடையாளம் காண கன்வல்யூஷன் நியூரல் நெட்வொர்க் (CNN) முறையை நாங்கள் முன்மொழிகிறோம். கேமரா படத்திலிருந்து மனித பணியின் கை அசைவுகளை அங்கீகரிப்பதே இதன் நோக்கம். CNNக்கான பயிற்சி மற்றும் சோதனைத் தரவைப் பெற கையின் நிலை மற்றும் நோக்குநிலை பயன்படுத்தப்படுகிறது. கை முதலில் ஒரு வடிகட்டி வழியாக அனுப்பப்படுகிறது மற்றும் வடிகட்டியைப் பயன்படுத்திய பிறகு, கை சைகைகளின் வகுப்பை முன்னறிவிக்கும் வகைப்படுத்தி வழியாக கையை அனுப்புகிறது. பின்னர் அளவீடு செய்யப்பட்ட படங்கள் CNN பயிற்சிக்கு பயன்படுத்தப்படுகின்றன.

TITLE OF CONTENT

LIST OF FIGURES

CHAPTER 1 – INTRODUCTION

1.1 GENERAL

American sign language is a predominant sign language Since the only disability D&M people have been communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. Deaf and dumb(D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language. In our project we basically focus on producing a model which can recognise Fingerspelling based hand gestures in order to form a complete word by combining each gesture. The gestures we aim to train are as given in the image below.

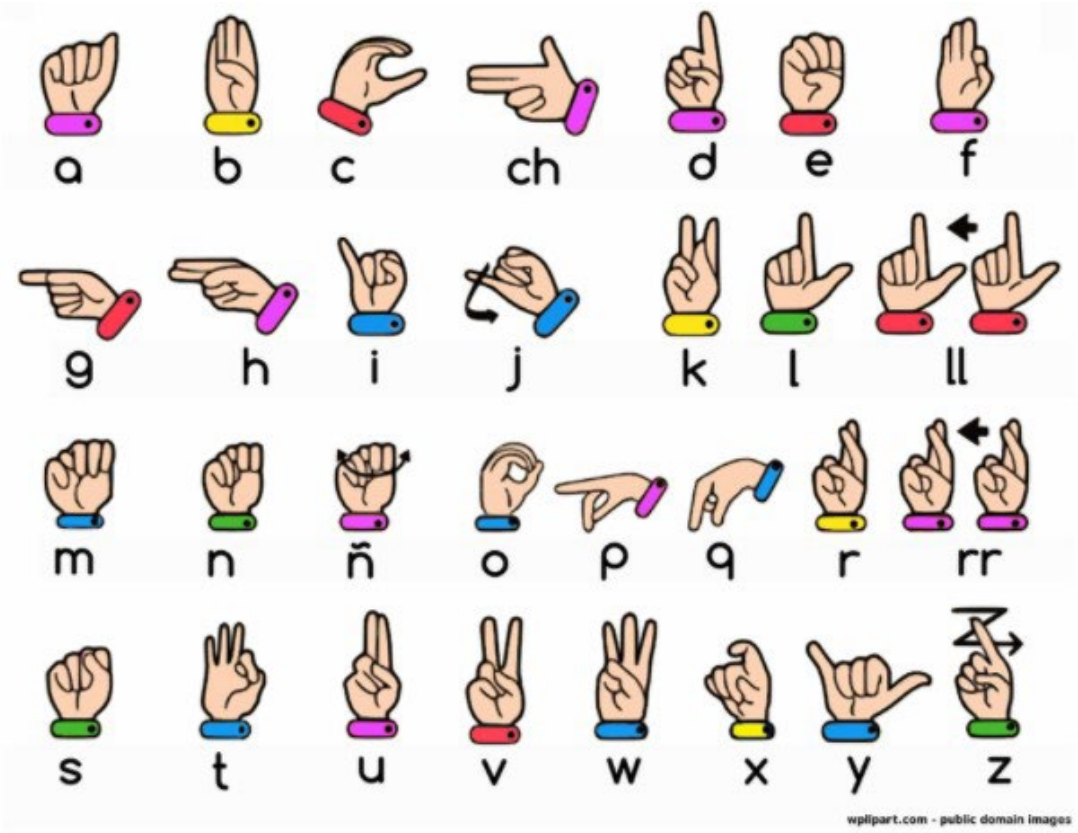


Fig.1.1 American Sign Language Gestures

1.2 PURPOSE

More than 70 million deaf people around the world use sign languages to communicate. Sign language allows them to learn, work, access services, and be included in the communities.

It is hard to make everybody learn the use of sign language with the goal of ensuring that people with disabilities can enjoy their rights on an equal basis with others.

So, the aim is to develop a user-friendly human computer interface (HCI) where the computer understands the American sign language This Project will help the dumb and deaf people by making their life easy.

1.3 SCOPE

This System will be Beneficial for Both Dumb/Deaf People and the People Who do not understands the Sign Language. They just need to do that with sign Language gestures and this system will identify what he/she is trying to say after identification it gives the output in the form of Text as well as Speech format.

1.4 MOTIVATION

Daily activities including Communication, Navigation, etc. are certainly particularly challenging tasks for visually impaired, blind, and deaf-mute people. Indoor navigation itself is certainly becoming a harder task for deaf-mute people.
At the principal sight, as a thought, how troublesome could make a sign dialects converter. We will catch capacities and specialized provisions to the motion catch of sign to voice Change.

The communication gap between the deaf and dumb people and the people who don't know sign language which will be bridged via and user interface which make conversations easier. We have taken the speech of the user as input and converted that input sentence to its corresponding sign language.
The way of transferring the information from one person to another is called communication. Most of the time people use signs and words for the communication. Natural language is used by normal people to communicate/interact with each other while tactile sign language is used by deaf and dumb people to interact.

PROBLEM STATEMENT

Nowadays people with disabilities experience difficulties to stand in the race because of ferocious competition in every field. The effort is to develop an interface which will help deaf and dumb people interact with a normal person. According to a survey, India consist of nearly 2.4 million deaf, dumb populations which approximately make up 20% of the world's total deaf and dumb population.
For hassle-free interaction between the normal person and deaf and dumb person, there is a need of an interpreter (Person who has the knowledge of sign language, as well as normal language).

Sign language is divided into two i.e. Visual Sign Language & Tactile Sign Language.

- Visual sign language: It is used by hearing & speech impaired people
- Tactile sign language: It is used by hearing & sight impaired people. We are basically working on the visual sign language used by deaf & dumb.

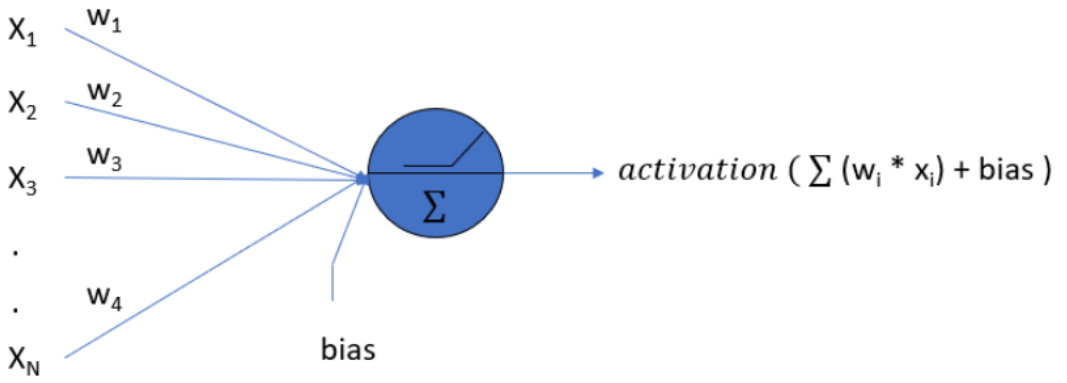
Sign Language varies country to country it depends on its culture as Sign language in India is ISL (Indian Sign Language), America uses ASL (American Sign Language), China uses CSL (Chinese Sign Language).
Sign Language is a method of communication for deaf & dumb which is composed of various gestures formed by hand shapes, body orientation & facial expression. Each gesture has a meaning assigned to it.
Alphabets in sign language are composed of different hand shapes & words are composed of hand shapes with orientation. Complete visual sign language also

includes facial expressions. Visual sign language is an effective means of communication for deaf & dumb.

Though it is true, the hearing-impaired have to challenge communication obstacles in a mostly hearing capable society

1.5 INTRODUCTION TO NEURAL NETWORK

Neural networks are used to mimic the basic functioning of the human brain and are inspired by how the human brain interprets information. It is used to solve various real-time tasks because of its ability to perform computations quickly and its fast responses.



A single neuron shown with X_i inputs with their respective weights W_i and a bias term and applied activation function

Fig.1.2 Neural Network Model

An Artificial Neural Network model contains various components that are inspired by the biological nervous system.

Artificial Neural Network has a huge number of interconnected processing elements, also known as Nodes. These nodes are connected with other nodes using a connection link. The connection link contains weights, these weights contain the information about the input signal. Each iteration and input in turn leads to updation of these weights. After inputting all the data instances from the training data set, the final weights of the Neural Network along with its architecture is known as the Trained Neural Network. This process is called Training of Neural Networks. This trained neural network is used to solve specific problems as defined in the problem statement.

Types of tasks that can be solved using an artificial neural network include Classification problems, Pattern Matching, Data Clustering, etc.

We use artificial neural networks because they learn very efficiently and adaptively. They have the capability to learn "how" to solve a specific problem from the training data it receives. After learning, it can be used to solve that specific problem very quickly and efficiently with high accuracy.

Some real-life applications of neural networks include Air Traffic Control, Optical Character Recognition as used by some scanning apps like Google Lens, Voice Recognition, etc.

Types of Neural Networks :

(i) ANN - It is also known as an artificial neural network. It is a feed-forward neural network because the inputs are sent in the forward direction. It can also contain hidden layers which can make the model even denser. They have a fixed length as specified by the programmer. It is used for Textual Data or Tabular Data. A widely used real-life application is Facial Recognition. It is comparatively less powerful than CNN and RNN.

(ii) CNN - It is also known as Convolutional Neural Networks. It is mainly used for Image Data. It is used for Computer Vision. Some of the real-life applications are object detection in autonomous vehicles. It contains a combination of convolutional layers and neurons. It is more powerful than both ANN and RNN.

(iii) RNN - It is also known as Recurrent Neural Networks. It is used to process and interpret time series data. In this type of model, the output from a processing node is fed back into nodes in the same or previous layers. The most known types of RNN are **LSTM** (Long Short Term Memory) Networks.

1.6 INTRODUCTION TO CNN

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

1.6.1 CNN architecture

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

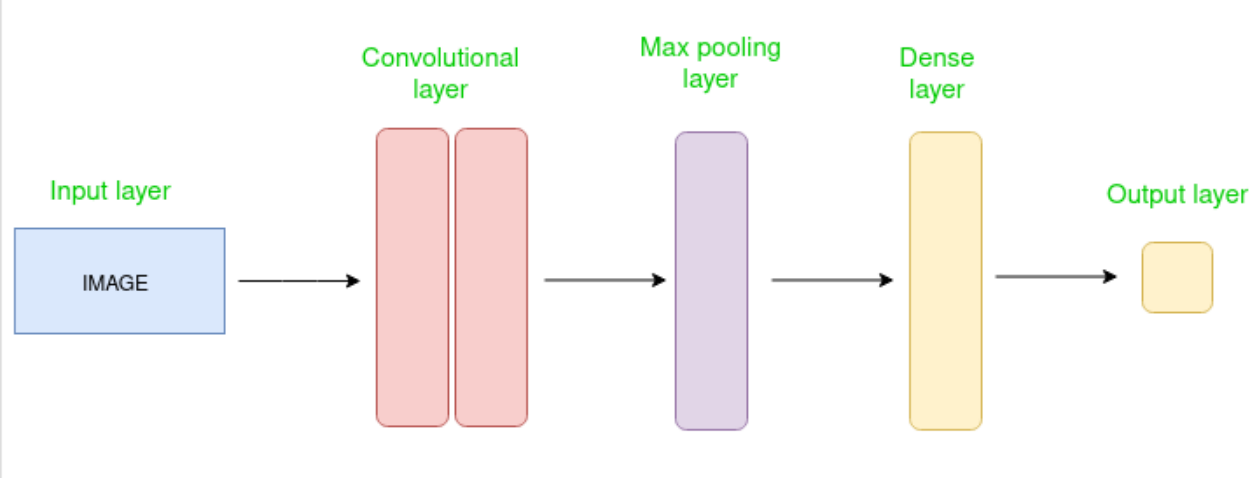


Fig.1.3 Simple CNN architecture

The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

1.6.2 How Convolutional Layers works

Convolution Neural Networks or convnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).

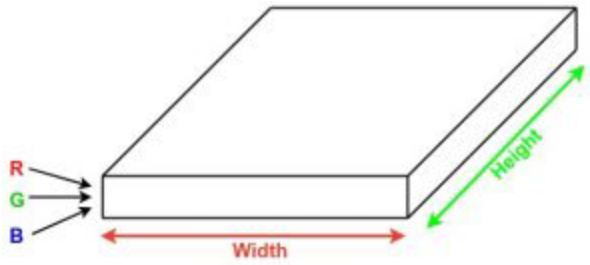


Fig 1.4 Image Channel

Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

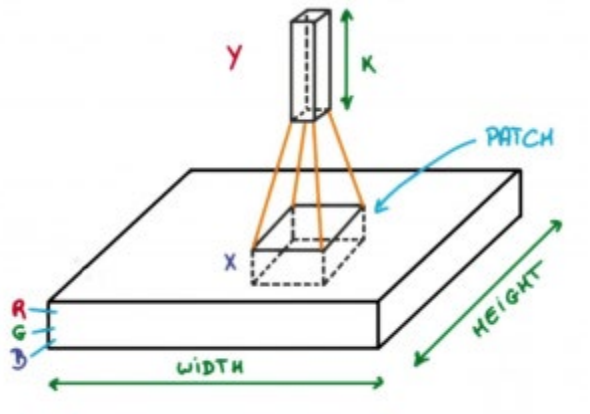


Fig 1.5 Convolution Operation in CNN

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.

- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

1.6.3 Layers used to build ConvNets

A complete Convolution Neural Networks architecture is also known as covnets.

A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

Types of layers:

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

- **Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

- **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred ad feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.

- **Activation Layer:** By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x), **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.

- **Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

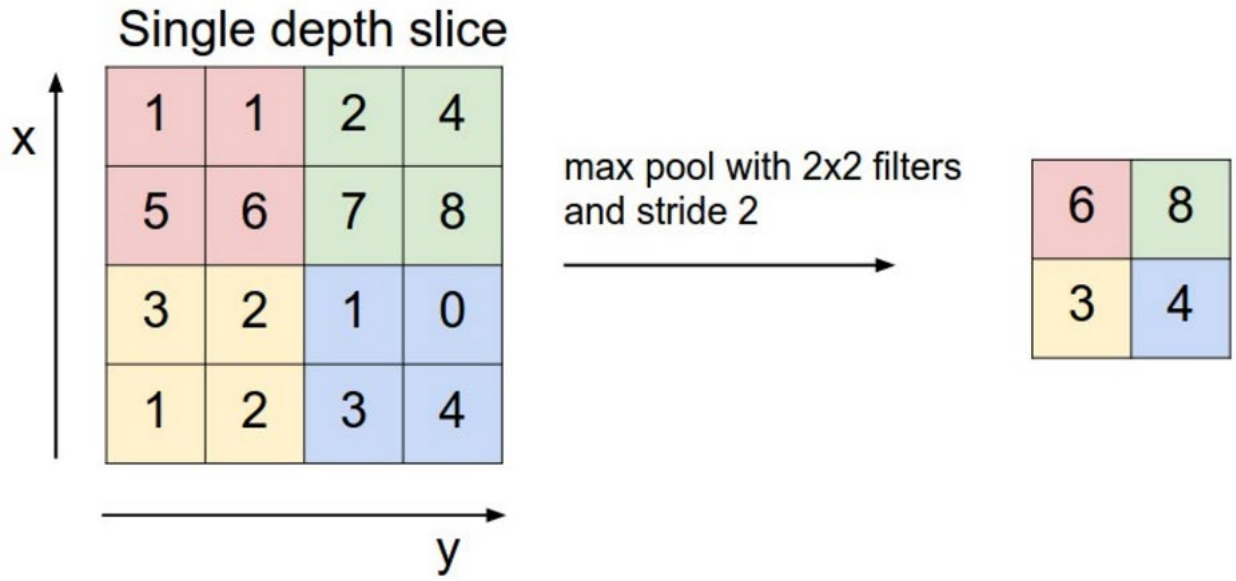


Fig 1.6 Max Polling

- **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.

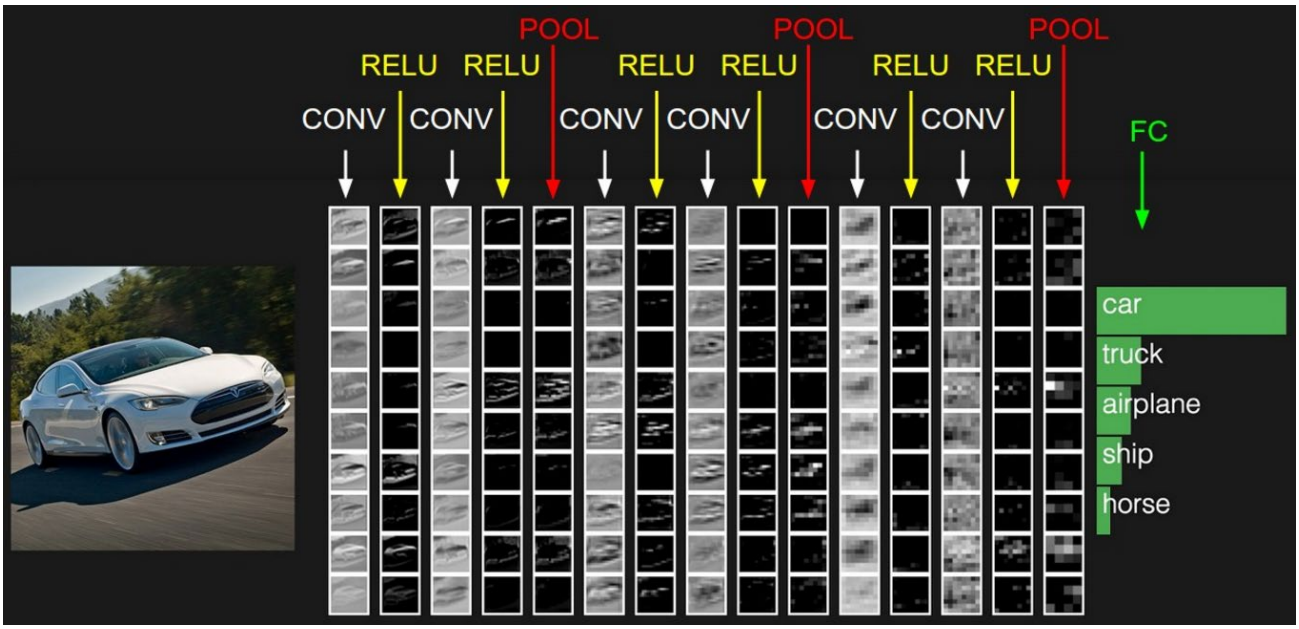


Fig 1.7 Fully Connected Layers

- **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class

CHAPTER 2 – LITERATURE SURVEY

Hearing-impaired/mute people can communicate more easily and naturally by using sign language. Researchers have become interested in sign language as a way to improve social communication among the deaf due to the rapid growth of multimedia communication systems [1].

Sensor or vision-based techniques were almost always used in sign language recognition systems. Vision-based uses either RGB or depth cameras to capture images and films in colour or depth. Video data is processed and classified using a variety of machine learning methods. Different sensors are included into an electromechanical glove to collect data in the sensor-based approach [2]. Numerous studies have been done to tackle different sign recognition assignments for different languages. [4] examined the recognition methods for the Chinese sign language. Xiao et al. [5] combined hand and skeletal sequence information using twin Long Short-Term Memory then a CHMM. Authority rate transformations and the fusion of RGB-D images are used to segment the hands. PCANet was used to features extracted from depth images in order to build alphabet recognition in American sign language.

A quiz-based learning device for finger-spelled sign in Indian sign language was created by Joy et al. [6]. Recognition of the Arabic language has recently been a major topic. Different features extraction methods such as DCT, Binary Classification (LBP) with HMM, and temporal LBP with SVM have been used to construct several separate Standard arabic gesture recognition systems. HMM was planned to distinguish solitary Arabic signs in [7], where a mixture of Stochastic skin modeling and a region-mounting approach were used to divide the face and hands. In Al-Rousan et al., Discrete Transformation (DCT) and Hidden Markov (HMMs) were used to recognise 30 solitary Arabic words (HMMs). Offline, online, and signer-dependent and signer-independent tests were undertaken by the researchers. There are usually three main processes in gesture recognition systems for isolated sign language gestures: hand segment, extraction of features and sequence classification. To excerpt hand areas from each frames of the participation video, these systems used a variety of hand segmentation approaches. An artificial neural network created by Dahmani [8] is able to separate the hand from a complex background using skin colour and texture features. Color gloves were used in [9] to segment the signer's hands and locate them on either side of his or her body. Finally, a multinomial classifier was utilised to categorize the separated Arabic signs using spatial coding choice of DCT coefficients. Many sophisticated computer vision problems can be solved more effectively using various deep learning architectures. For static or dynamic sign language recognition, a variety of deep neural networks have recently

been applied. Molchanov et al. [10] used a convolutional neural network to simultaneously detect and identify dynamic hand gestures from depth, colour, and stero-IR sensor data.

Recurrent neural networks are trained to automatically extract three-dimensional and temporal data from brief clips, which are input into a 3D-CNN. The current clip's hidden state is fed into a softmax layer that uses productive harmony classification as a cost function to calculate class conditional probabilities. Convolutional neural networks and bidirectional LSTM networks have been created by Liao et al. [11] for dynamical sign language recognition.

Localizing the hand object using R-CNN, a 3dimensional ResNet simultaneously gathers spatial and temporal data from the video clips, which are categorized using bidirectional LSTM bidirectional LSTM classification algorithm There is an attention networks with feature space (LS-HAN) structure for continual sign recognition developed by Jie and colleagues [12]. Word temporal segmentation was a goal of this approach. Two-stream Convolutional Neural Networks (CNN) are used to represent video features, Latent Space (LS) is used to bridge the gaps, and a Hierarchy Attention Network (HAN) is used to recognise objects in the Latent Space (LS). Using attention-based 3D-convolutional neural networks, Huang et al (3D-CNNs). Attention mechanisms let the model focus on areas of interest by allowing it to learn from raw video. After the features were extracted, we used temporal attention to pick out the most noteworthy motions for further analysis. Stacking temporal fusion and bi-directional recursive neural networks modules were used by Cui for extracting features and sequence modelling, respectively. The end-to-end framework for the sequence alignment proposal was trained using an iterative optimization technique. The feature extraction was fine-tuned with the alignment suggestion as supervisory information, resulting in improved network performance. Use of the Convolutional Neural Network for hand categorization and activity recognition has recently gained significant success (CNN). Convolutional neural networks CNN's like Semantic segmentation [13], RefineNet, U-Net, and DeepLab have shown remarkable performance in a wide range of object segmentation challenges. With the use of a set of skin-based bounding boxes, Bambach et al. used CNN and GrabCut to recognize and segment the hand. Using in-layer multiscale neurons, an intensity deep neural network was developed in [14]. Deep neural networks were used in [15] to segment the hands, and they were fine-tuned. The RefineNet's hand maps were then utilised to recognise hand activity. It was created by Wang et al. [16] to operate in an asset environment with a restricted processing capacity.

CHAPTER 3 – SYSTEM ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

In order for every software application to run properly, it needs to satisfy a lot of functions that are to be deployed in it. These functions are nothing but various operations that are performed in each step while developing the application. This step comes under the best practices of developing an application. Functional and Non-Functional Requirements together set a list of rules that govern the smooth running of an application and it also helps the developer and the user to determine the software and hardware requirements that are needed to run the application. Functional Requirements that are required are:

Python:

Python programming language was developed in the year 1991 by Guido Van Rossum. The syntaxes used in the language makes it very comfortable and easier for developers to work with. Because of this very reason, this programming language can be used both in small and large scale. They are dynamic and garbage collected.

OpenCV:

OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java, C++.

Numpy:

Numpy is a universally useful array processing package. It gives an elite multidimensional cluster object, and devices for working with these arrays. It is the principal package for logical processing with Python.

Keras:

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

Mediapipe:

The MediaPipe framework is mainly used for rapid prototyping of perception pipelines with AI models for inferencing and other reusable components. It also facilitates the deployment of computer vision applications into demos and applications on different hardware platforms.

Tensorflow:

The TensorFlow platform helps you implement best practices for data automation, model tracking, performance monitoring, and model retraining. Using production-level tools to automate and track model training over the lifetime of a product, service, or business process is critical to success.

3.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are used to set conditions to monitor the performance characteristic of the application. It describes how a specific function in the application works. They also determine the overall quality of the project and hence it is a very important aspect in any software development process. The Non-Functional Requirements include

1. **Usability:** It refers to the easiness of the application of models and determines the ease with which it can be used by the user. Usability can be said to be high when the knowledge required to use the models is less and the efficiency of its functionality is high. It is also a main criterion which can determine the accuracy of the results.

2. **Accuracy:** Accuracy determines the relative closeness of the value produced by the system to that of the ideal value. It is also one way to determine how the classification models works better compared to the other similar models.

3. **Responsiveness:** Responsiveness is determined by completing the software operations with minimal errors or no errors. It is directly proportional to the stability and the performance of the application. The Robustness and Recoverability can also be determined by this criterion.

4. **Scalability:** Scalability is used to determine the growth of the project. It determines how much room the application can have in order to include more features in the future. It determines the sustainability of the project.

3.3 HARDWARE REQUIREMENTS

Processor: Intel I5 processor

Storage Space: 500 GB.

Screen size: 15” LED

Devices Required: Monitor, Mouse, Keyboard and a WebCam

Minimum Ram: 8GB

3.4 SOFTWARE REQUIREMENTS

Operating System: Windows 8 and Above

Programming Language: Python 3.9 5

IDE: PyCharm

Python libraries: OpenCV, NumPy, Keras, Mediapipe, Tensorflow.

3.5 ISSUES IN EXISTING SYSTEM

Systems exist where Neural Networks are used to detect Sign Language. However these require more computational power which cannot be carried out by, say a handheld device like a mobile phone or a Tablet. Further, some other systems require a High-Tech glove with motion sensors to capture the gesture in 3 dimensions or even a Microsoft Kinect, both of which are pretty International Journal of Pure and Applied Mathematics Special Issue 1689 expensive. These also impose scalability issues due to the equipment dependency. Benefits of portability has to be compromised, with the equipment having to be carried everywhere where the system is to be used. The Microsoft Kinect is the other popular method used by researchers to acquire their data. Microsoft Kinect is getting more popular among researchers as it provides more data and it is needed by researchers and uses Kinect to acquire their data. the advantages of using Kinect is that it provides the depth data of the video stream. The depth data is very useful as it can easily distinguish the background and the signer. Furthermore, it can be used to distinguish hands and body as the signer usually perform sign language by hands in front of their body. The disadvantage is that the Microsoft Kinect device is costly and it should be connected to computer. Uses simple camera and color gloves to differentiate both hands and ease the feature extraction process. Using 3-axis accelerometer and flex sensors; uses 5-Dimensional Tracker; while 18 uses accelerometer (ACC) and surface electromyography (sEMG). Uses 2 Cybergloves and 3 SPACE-position tracker. All of these gloves are equipped with sensors attached to the gloves. The advantage is that it provides all the data needed more accurately as it also provides fingers movement data. The disadvantages are that they are costly and are difficult to be used commercially.

CHAPTER 4 – SYSTEM DESIGN

4.1 SYSTEM FLOW CHART

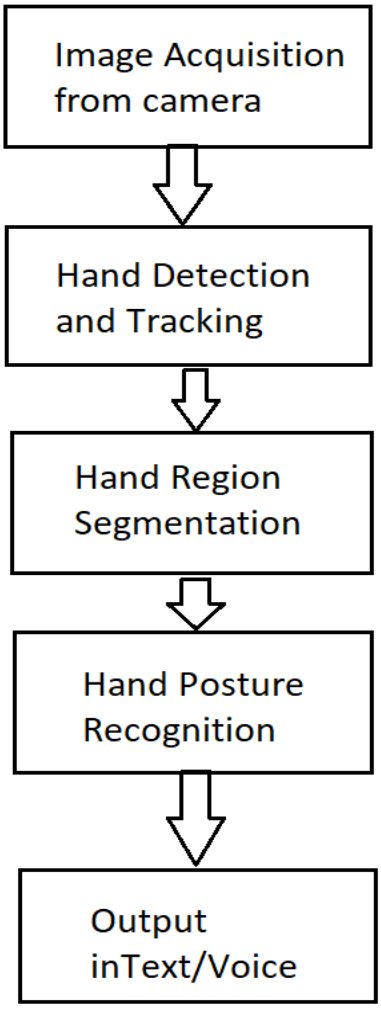


Fig 4.1: Proposed System Flowchart

System flowcharts are the diagram type that shows you the flow of data and how decisions can affect the events surrounding it. Like other types of flowcharts, system flowcharts consist of start/end terminals, processes, and decisions, all connected by arrows showing the flow and how data moves in the flow.

4.2 SYSTEM USE CASE DIAGRAM

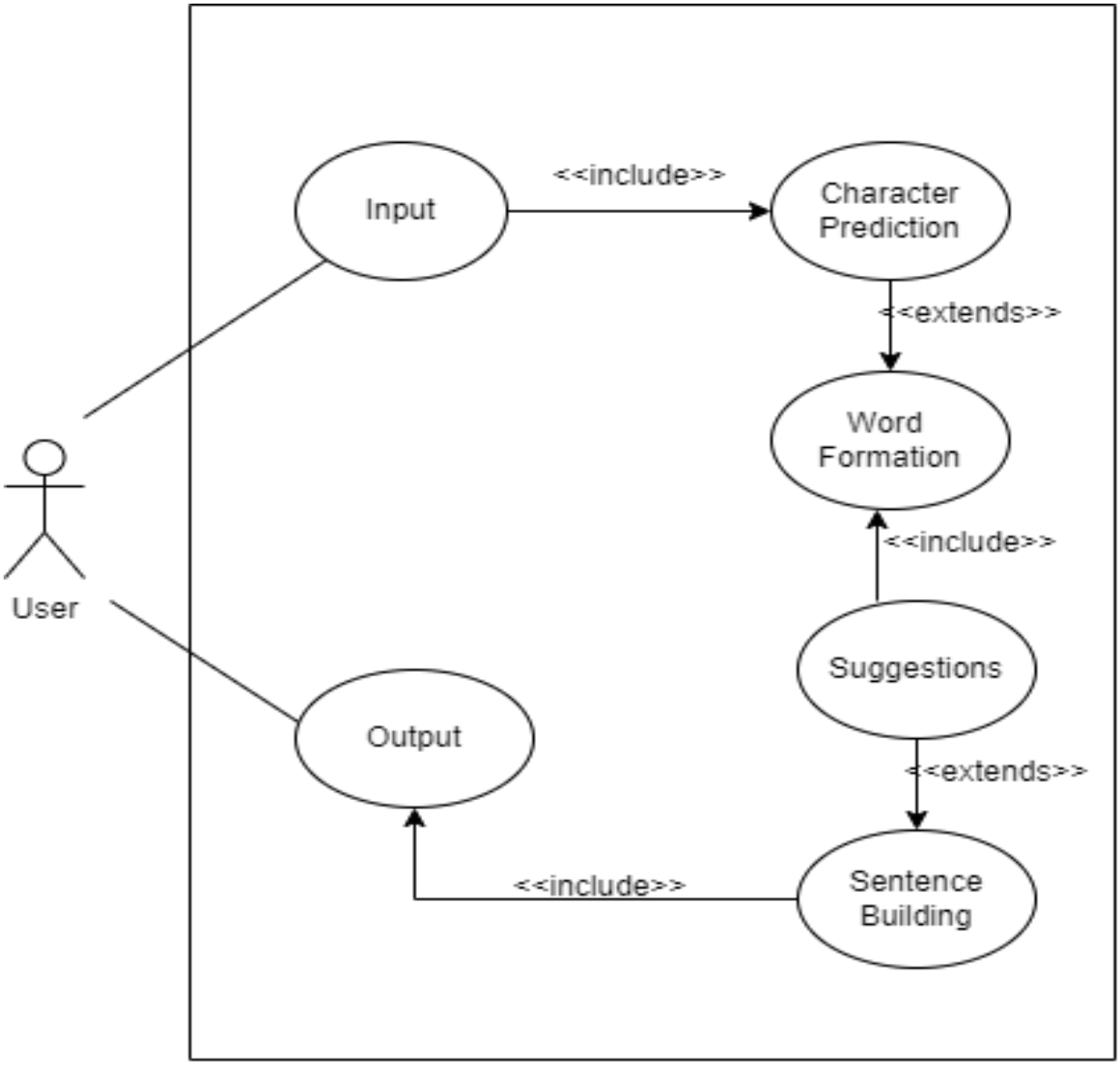


Fig 4.2 : Use Case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

4.3 SYSTEM DATA FLOW DIAGRAM

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

Level 1 DFD: This is the highest-level DFD, which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes.

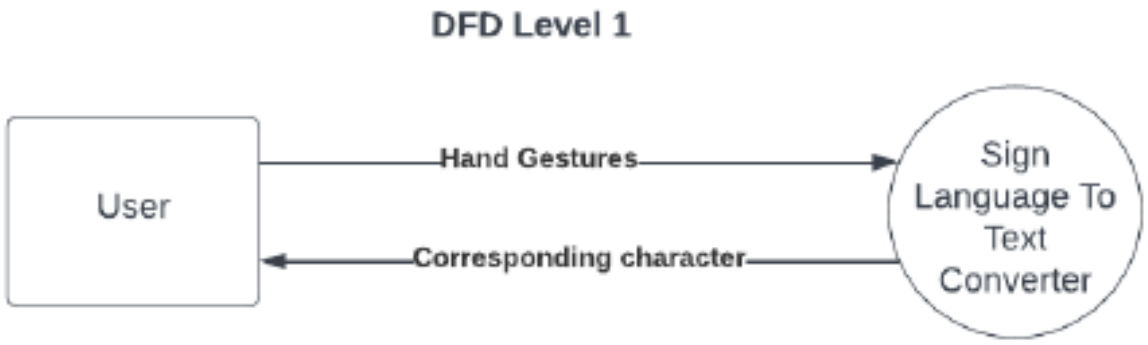


Fig 4.3 : DFD Level 1

Level 2 DFD: This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes. Each sub-process is depicted as a separate process on the level 2 DFD. The data flows and data stores associated with each sub-process are also shown.

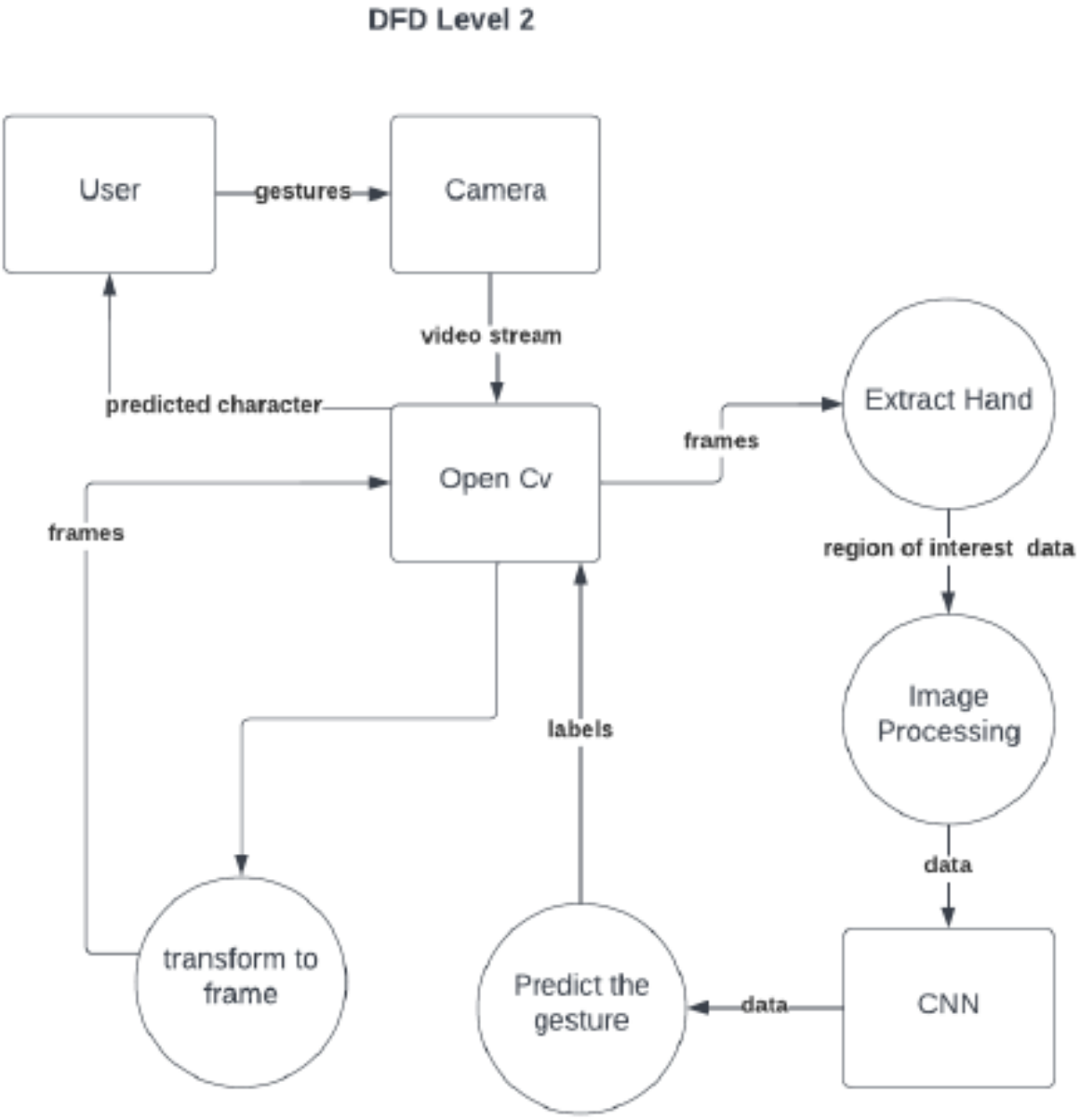


Fig 4.4 : DFD Level 2

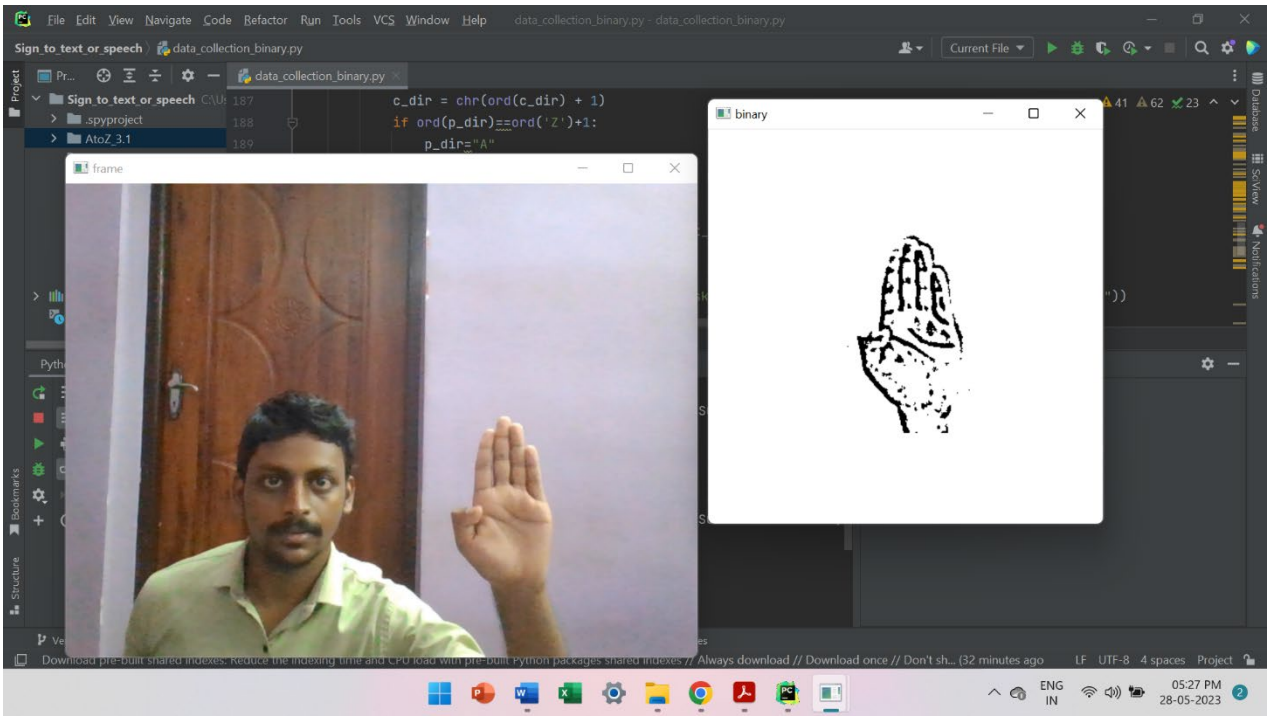
CHAPTER 5 – PROPOSED METHODOLOGY

5.1 PROCURING THE DATASET

The different approaches to acquire data about the hand gesture can be done in the following ways:

It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

In vision-based methods, the computer webcam is the input device for observing the information of hands and/or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices, thereby reducing costs. The main challenge of vision-based hand detection ranges from coping with the large variability of the human hand's appearance due to a huge number of hand movements, to different skin-color possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.



5.1.1 Data pre-processing and Feature extraction

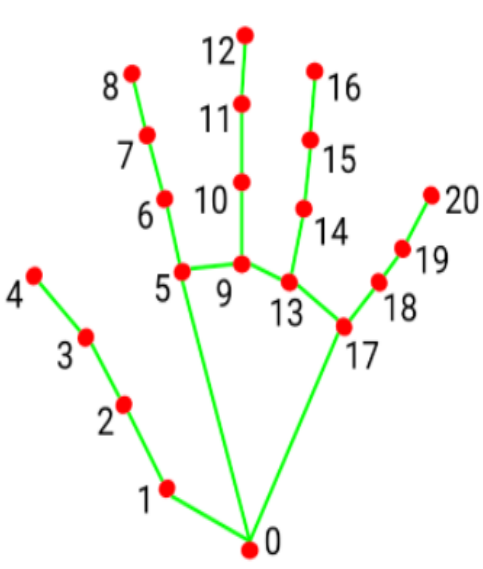
In this approach for hand detection, firstly we detect hand from image that is acquired by webcam and for detecting a hand we used media pipe library which is used for image processing. So, after finding the hand from image we get the region of interest (Roi) then we cropped that image and convert the image to gray image using OpenCV library after we applied the gaussian blur .The filter can be easily applied using open computer vision library also known as OpenCV. Then we converted the gray image to binary image using threshold and Adaptive threshold methods.

We have collected images of different signs of different angles for sign letter A to Z.

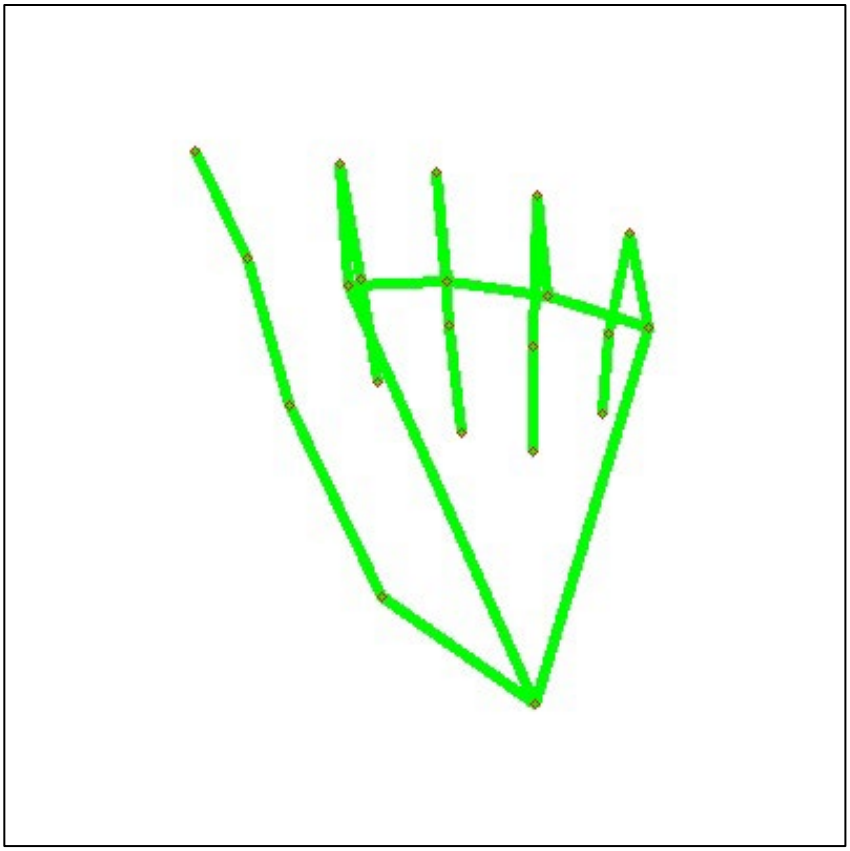
In this method there are many loop holes like your hand must be ahead of clean soft background and that is in proper lightning condition then only this method will give good accurate results but in real world we dont get good background everywhere and we don't get good lightning conditions too.

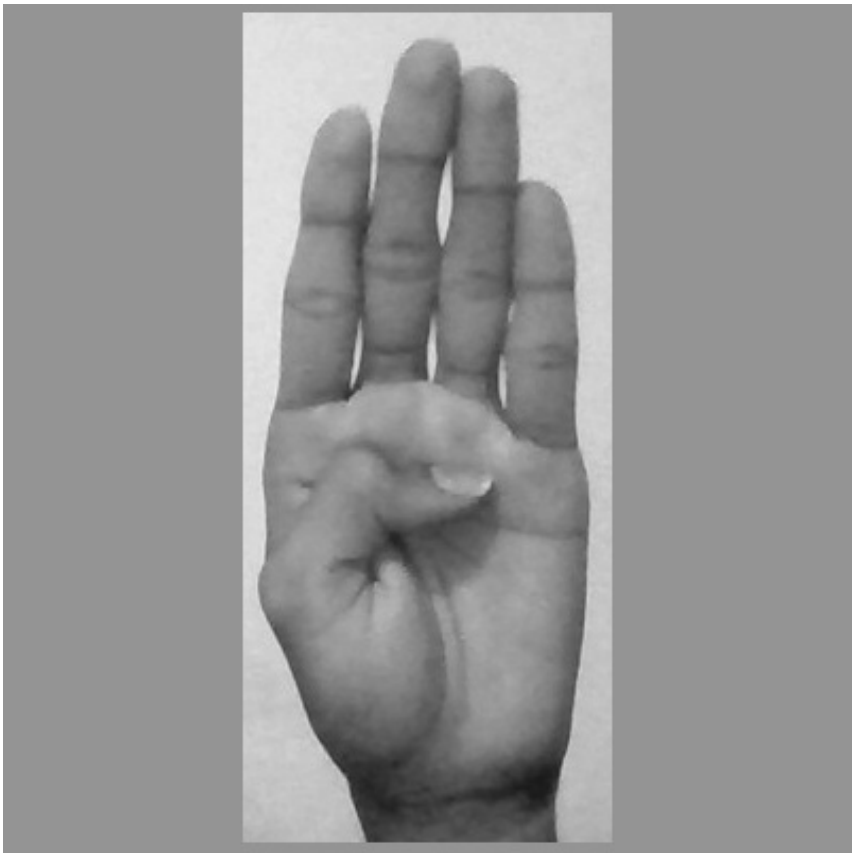
So, to overcome this situation we try different approaches then we reached at one interesting solution in which firstly we detect hand from frame using mediapipe and get the hand landmarks of hand present in that image then we draw and connect those landmarks in simple white image.

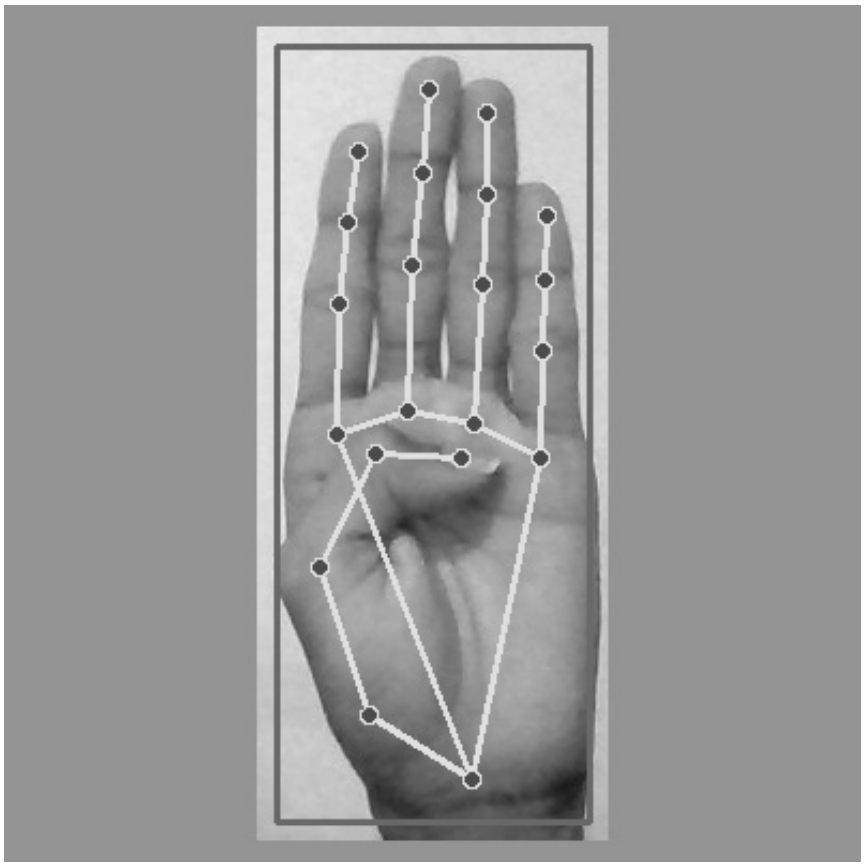
5.1.2 Mediapipe Landmark System

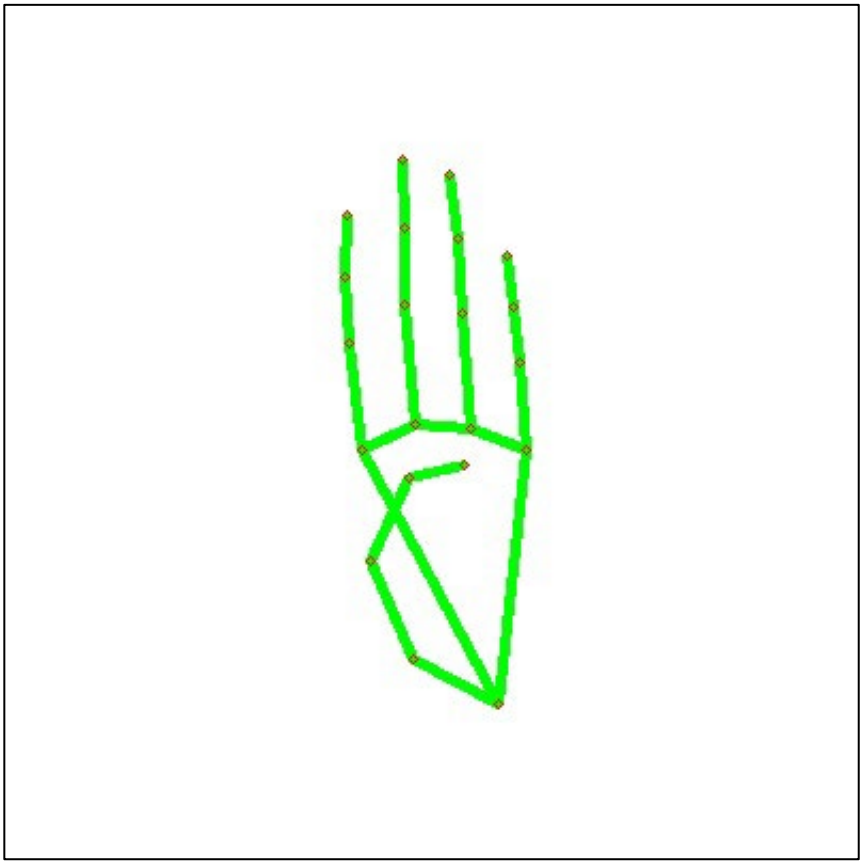


0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP



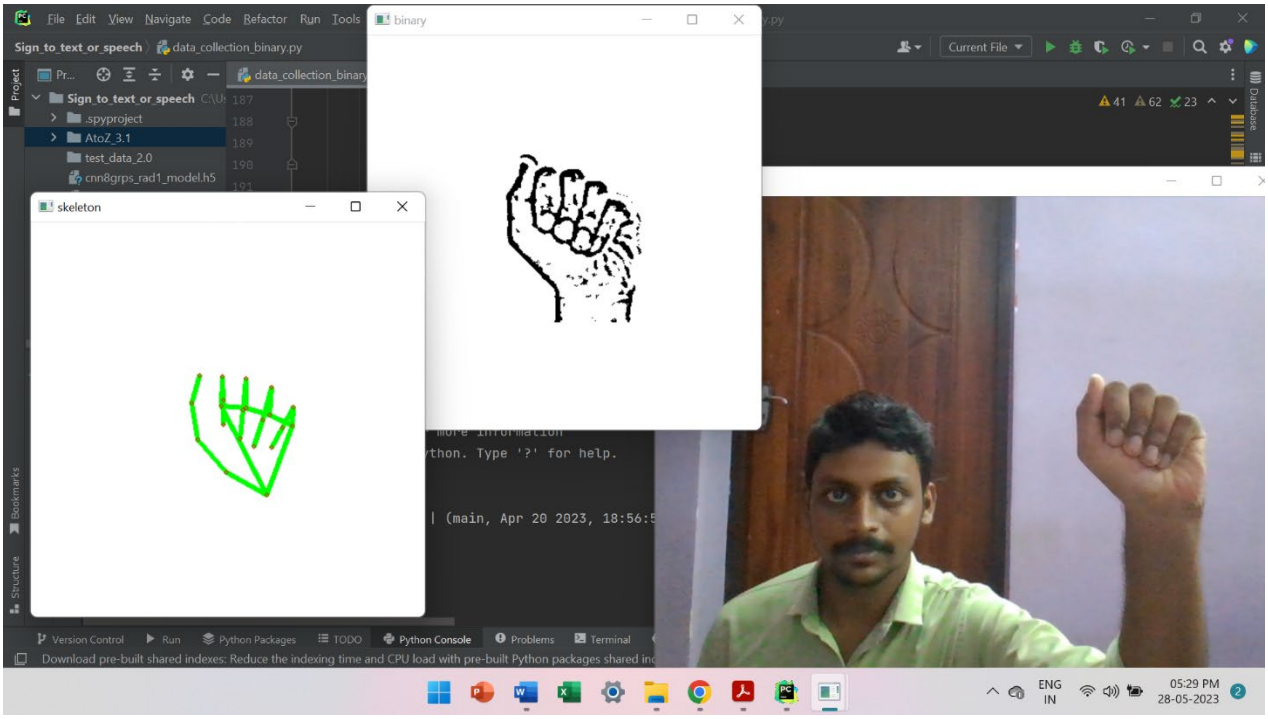


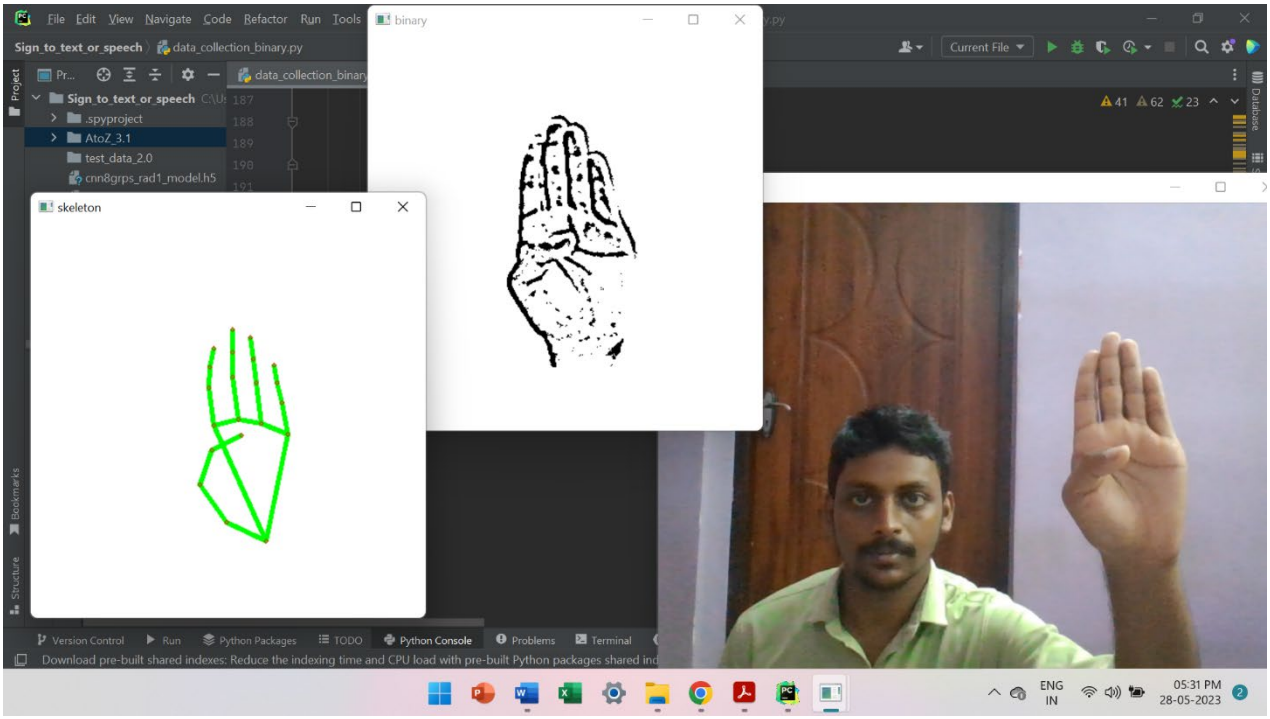


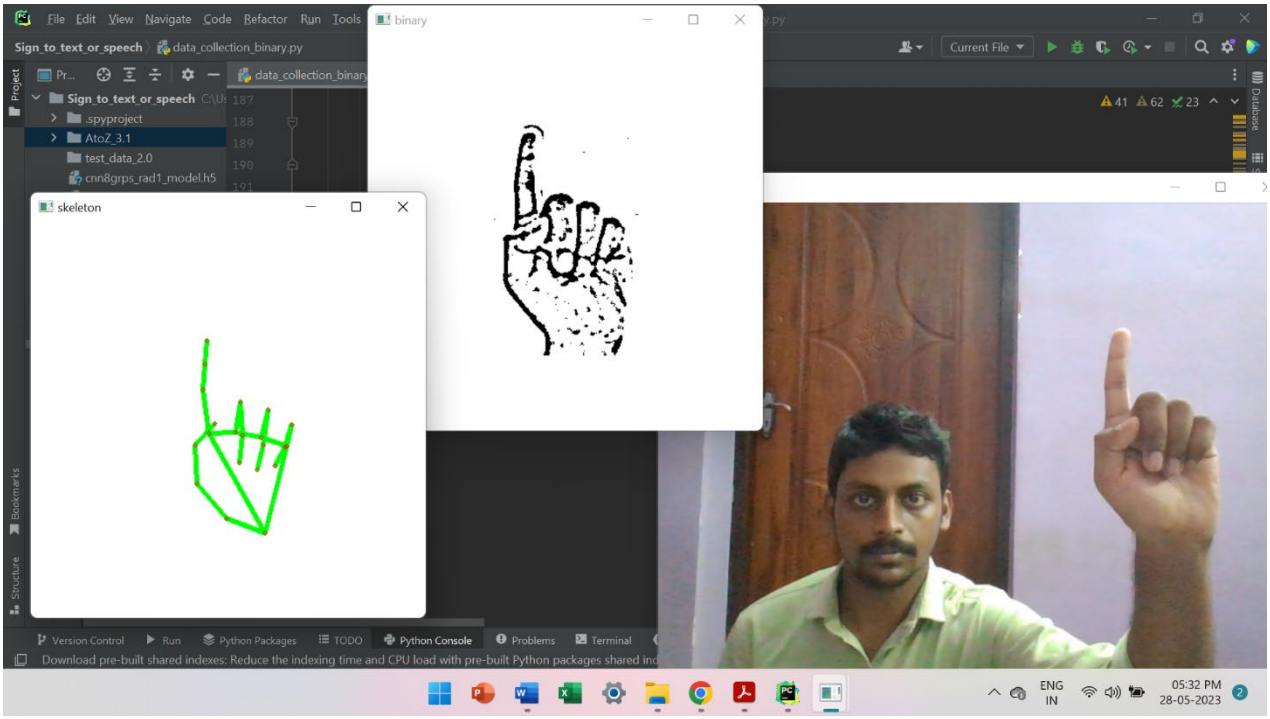


Now we get this landmark points and draw it in plain white background using opencv library

-By doing this we tackle the situation of background and lightning conditions because the mediapipe library will give us landmark points in any background and mostly in any lightning conditions.







-we have collected 180 skeleton images of Alphabets from A to Z

5.2 GESTURE CLASSIFICATION

5.2.1 Convolutional Neural Network (CNN)

CNN is a class of neural networks that are highly useful in solving computer vision problems. They found inspiration from the actual perception of vision that takes place in the visual cortex of our brain. They make use of a filter/kernel to scan through the entire pixel values of the image and make computations by setting appropriate weights to enable detection of a specific feature. CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful tool that can identify features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth.

The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner.

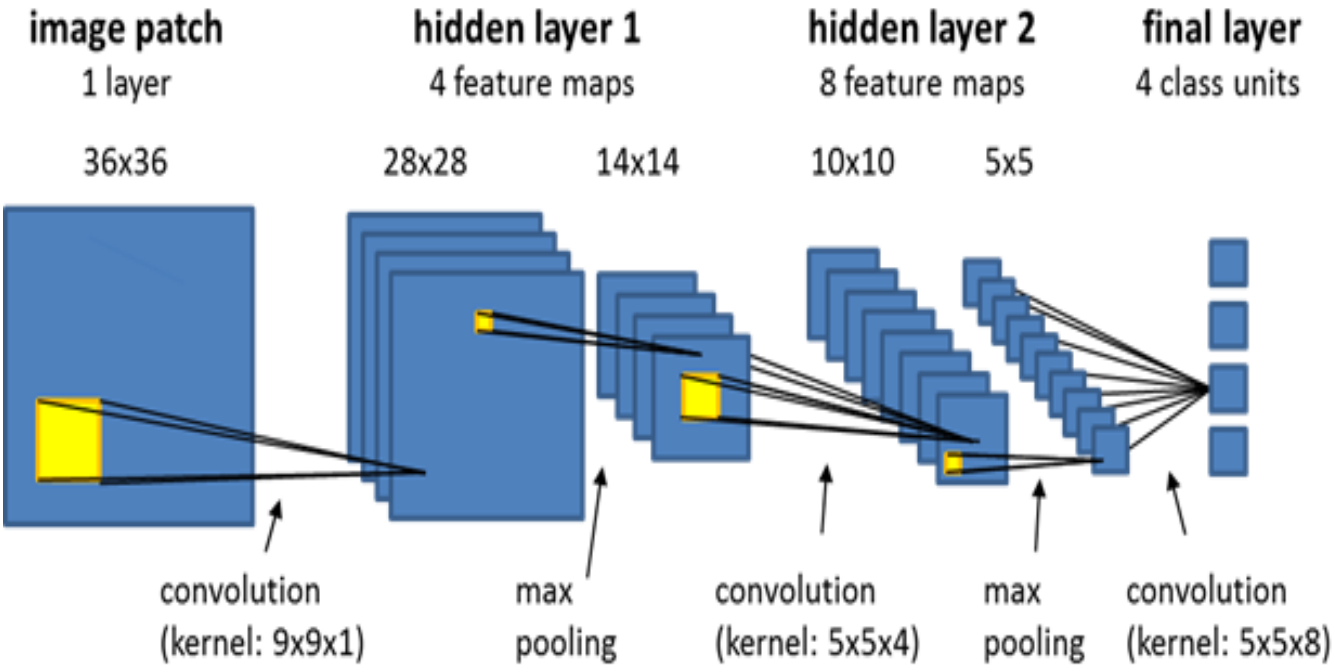
Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

5.2.2 Convolutional Layer

In convolution layer I have taken a small window size [typically of length 5*5] that extends to the depth of the input matrix.

The layer consists of learnable filters of window size. During every iteration I slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.

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



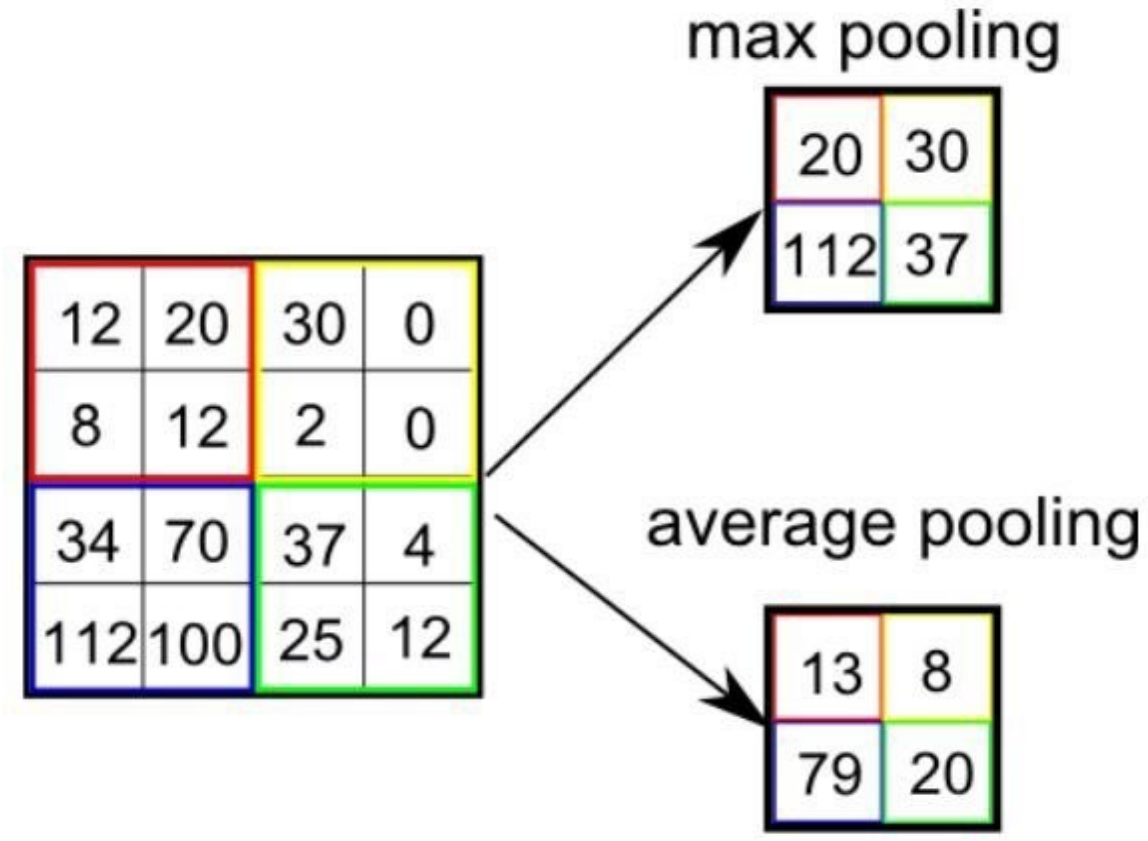
5.2.3 Pooling Layer

We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters.

There are two types of pooling:

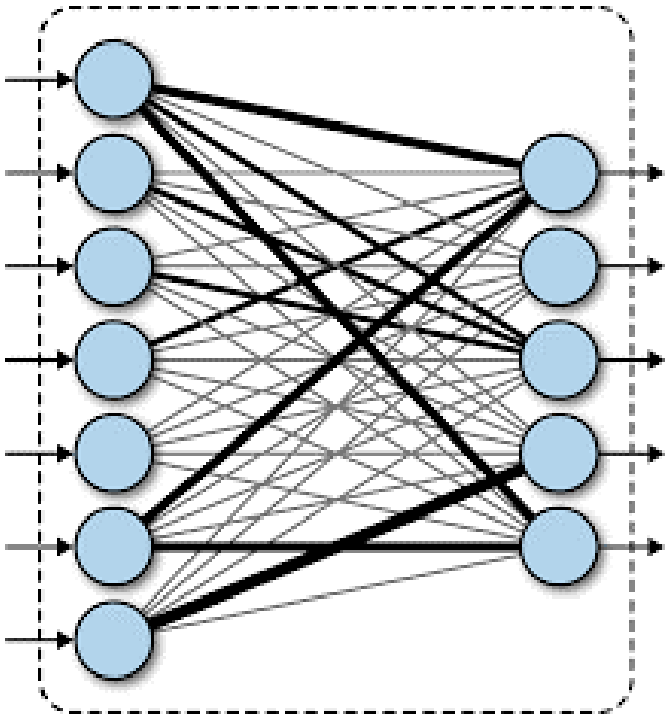
a. Max Pooling:
In max pooling we take a window size [for example window of size 2*2], and only taken the maximum of 4 values. Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.

b. Average Pooling:
In average pooling we take average of all Values in a window.
pooling.



5.2.4 Fully Connected Layer

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



The preprocessed 180 images/alphabet will feed the keras CNN model.
Because we got bad accuracy in 26 different classes thus, We divided whole 26 different alphabets into 8 classes in which every class contains similar alphabets:

[y,j]

[c,o]

[g,h]

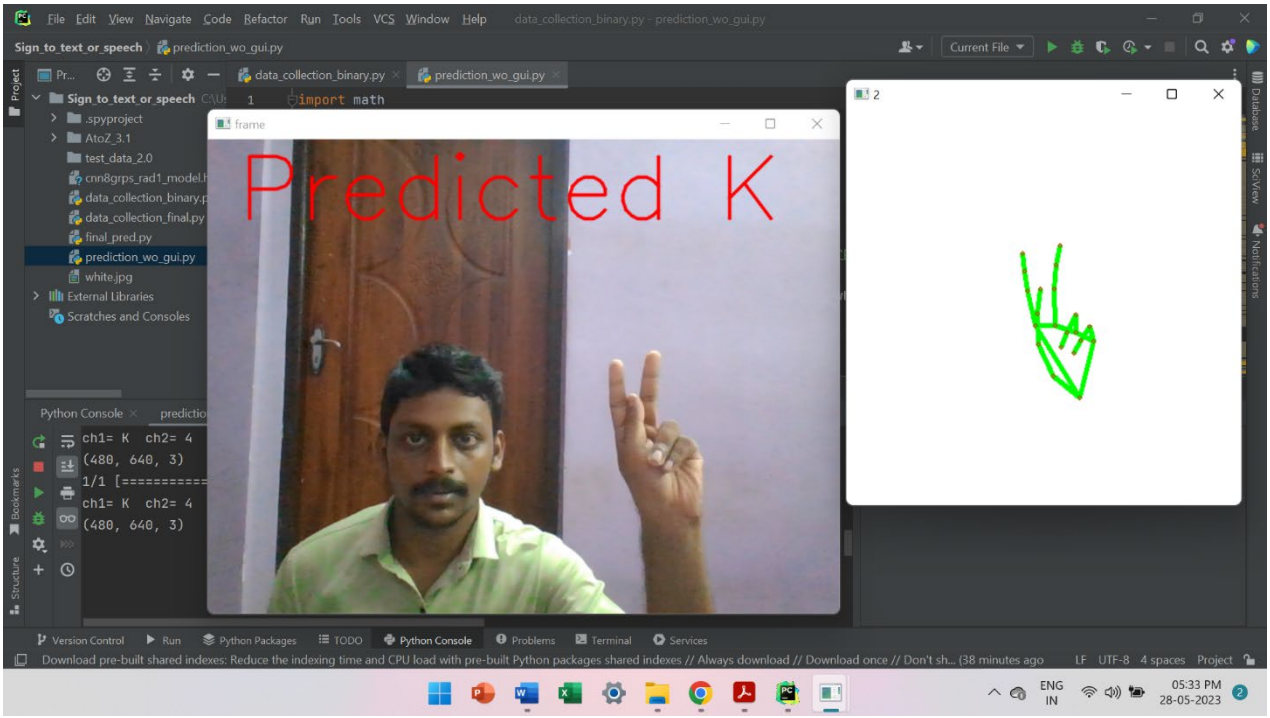
[b,d,f,I,u,v,k,r,w]

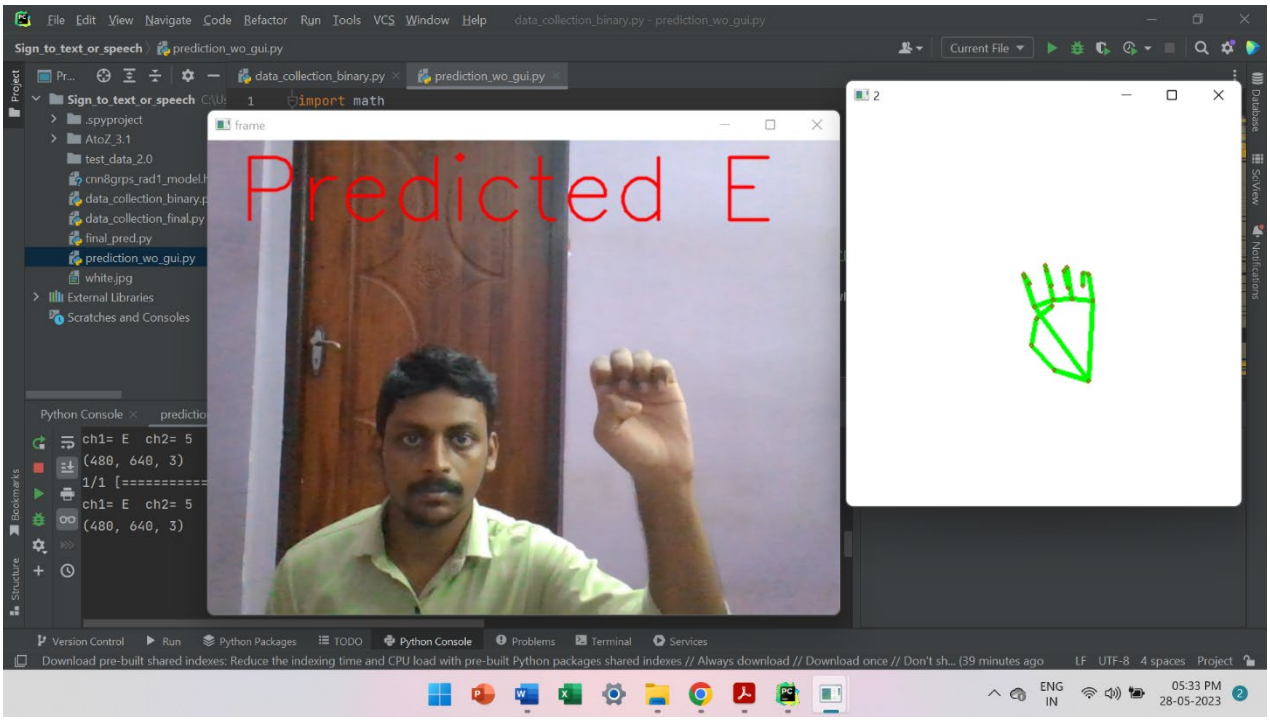
[p,q,z]

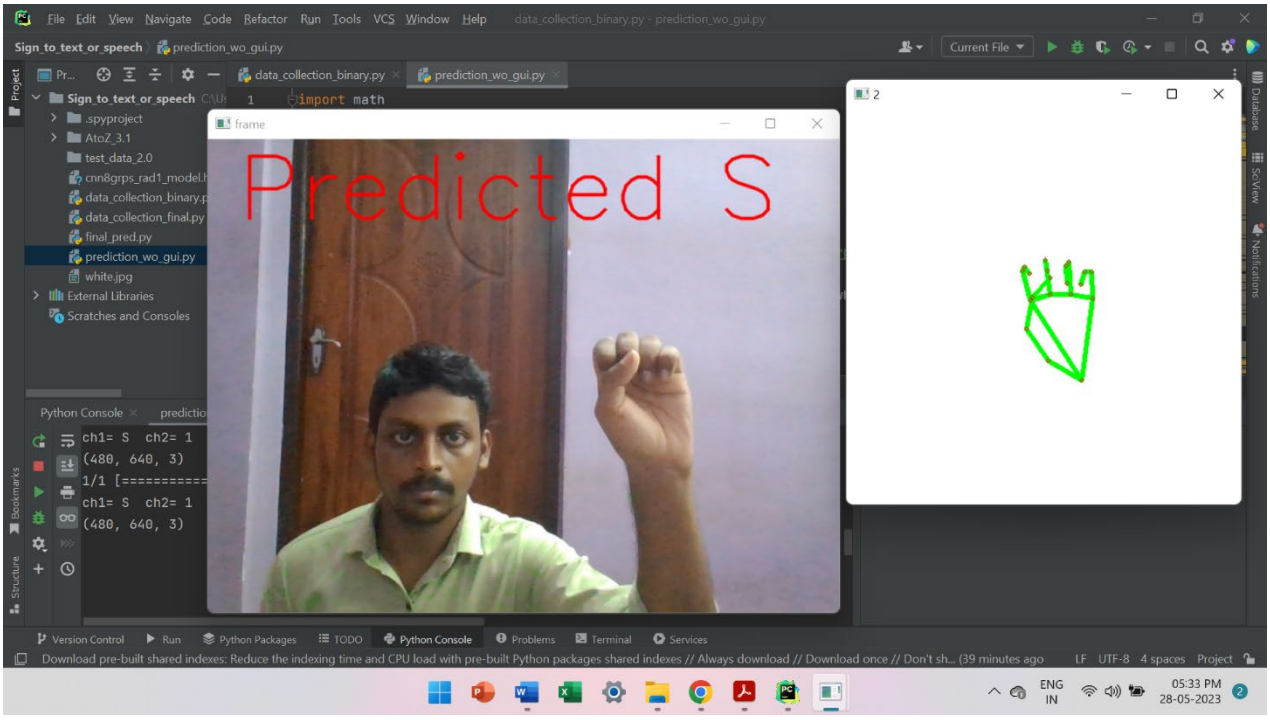
[a,e,m,n,s,t]

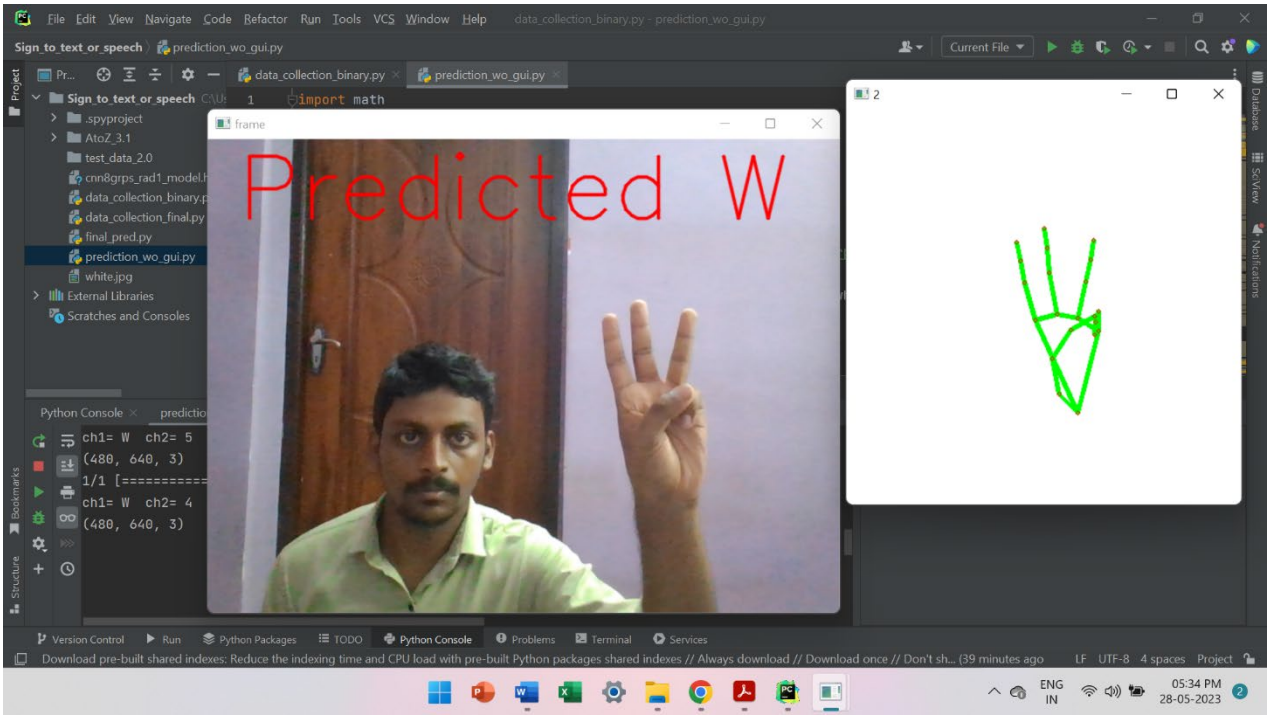
All the gesture labels will be assigned with a probability. The label with the highest probability will treated to be the predicted label. So when model will classify [aemnst] in one single class using mathematical operation on hand landmarks we will classify further into single alphabet a or e or m or n or s or t.

-Finally, we got **97%** Accuracy (with and without clean background and proper lightning conditions) through our method. And if the background is clear and there is good lightning condition then we got even **99%** accurate results.




Predicted E
ch1= E ch2= 5
(480, 640, 3)
ch1= E ch2= 5
(480, 640, 3)


Predicted S
ch1= S ch2= 1
(480, 640, 3)
ch1= S ch2= 1
(480, 640, 3)



Text To Speech Translation:

The model translates known gestures into words. we have used pyttsx3 library to convert the recognized words into the appropriate speech. The text-to-speech output is a simple workaround, but it's a useful feature because it simulates a real-life dialogue.

CHAPTER 8 – CODE

CODES FOR THE MAIN PROGRAM

```
# Importing Libraries
import numpy as np
import math
import cv2
import os, sys
import traceback
import pyttsx3
from keras.models import load_model
from cvzone.HandTrackingModule import HandDetector
from string import ascii_uppercase
import enchant
ddd=enchant.Dict("en-US")
hd = HandDetector(maxHands=1)
hd2 = HandDetector(maxHands=1)
import tkinter as tk
from PIL import Image, ImageTk

offset=29

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


# Application :

class Application:

    def __init__(self):
        self.vs = cv2.VideoCapture(0)
        self.current_image = None
        self.model =
load_model("C:/Users/Hp/Desktop/Sign_to_text_or_speech/cnn8grps_rad1_model.h
5")
        self.speak_engine=pyttsx3.init()
        self.speak_engine.setProperty("rate",100)
        voices=self.speak_engine.getProperty("voices")
        self.speak_engine.setProperty("voice",voices[0].id)
```

```
        self.ct = {}
        self.ct['blank'] = 0
        self.blank_flag = 0
        self.space_flag=False
        self.next_flag=True
        self.prev_char=""
        self.count=-1
        self.ten_prev_char=[]
        for i in range(10):
            self.ten_prev_char.append(" ")


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

        print("Loaded model from disk")

        self.root = tk.Tk()
        self.root.title("Sign Language To Text And Speech Conversion")
        self.root.protocol('WM_DELETE_WINDOW', self.destructor)
        self.root.geometry("1920x1080")

        self.panel = tk.Label(self.root)
        self.panel.place(x=50, y=3, width=730, height=540)

        self.panel2 = tk.Label(self.root)  # initialize image panel
        self.panel2.place(x=700, y=25, width=520, height=520)

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

        self.panel3 = tk.Label(self.root)  # Current Symbol
        self.panel3.place(x=280, y=520)

        self.T1 = tk.Label(self.root)
        self.T1.place(x=10, y=520)
        self.T1.config(text="Character :", font=("Courier", 30, "bold"))

        self.panel5 = tk.Label(self.root)  # Sentence
        self.panel5.place(x=260, y=575)
```

```
        self.T3 = tk.Label(self.root)
        self.T3.place(x=10, y=575)
        self.T3.config(text="Sentence :", font=("Courier", 30, "bold"))

        self.T4 = tk.Label(self.root)
        self.T4.place(x=10, y=630)
        self.T4.config(text="Suggestions :", fg="red", font=("Courier", 30, "bold"))


        self.b1=tk.Button(self.root)
        self.b1.place(x=390,y=630)

        self.b2 = tk.Button(self.root)
        self.b2.place(x=590, y=630)

        self.b3 = tk.Button(self.root)
        self.b3.place(x=790, y=630)

        self.b4 = tk.Button(self.root)
        self.b4.place(x=990, y=630)

        self.speak = tk.Button(self.root)
        self.speak.place(x=1100, y=500)
        self.speak.config(text="Speak", font=("Courier", 20), wraplength=100,
command=self.speak_fun)

        self.clear = tk.Button(self.root)
        self.clear.place(x=950, y=500)
        self.clear.config(text="Clear", font=("Courier", 20), wraplength=100,
command=self.clear_fun)


        self.str = " "
        self.ccc=0
        self.word = " "
        self.current_symbol = "C"
        self.photo = "Empty"


        self.word1=" "
        self.word2 = " "
        self.word3 = " "
        self.word4 = " "
```

```
        self.video_loop()

    def video_loop(self):
        try:
            ok, frame = self.vs.read()
            cv2image = cv2.flip(frame, 1)
            hands = hd.findHands(cv2image, draw=False, flipType=True)
            cv2image_copy=np.array(cv2image)
            cv2image = cv2.cvtColor(cv2image, cv2.COLOR_BGR2RGB)
            self.current_image = Image.fromarray(cv2image)
            imgtk = ImageTk.PhotoImage(image=self.current_image)
            self.panel.imgtk = imgtk
            self.panel.config(image=imgtk)

            if hands:
                # #print(" --------- lmlist=",hands[1])
                hand = hands[0]
                x, y, w, h = hand['bbox']
                image = cv2image_copy[y - offset:y + h + offset, x - offset:x + w + offset]

                white =
cv2.imread("C:/Users/Hp/Desktop/Sign_to_text_or_speech/white.jpg")
                # img_final=img_final1=img_final2=0

                handz = hd2.findHands(image, draw=False, flipType=True)
                print(" ", self.ccc)
                self.ccc += 1
                if handz:
                    hand = handz[0]
                    self.pts = hand['lmList']
                    # x1,y1,w1,h1=hand['bbox']

                    os = ((400 - w) // 2) - 15
                    os1 = ((400 - h) // 2) - 15
                    for t in range(0, 4, 1):
                        cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                                 (0, 255, 0), 3)
                    for t in range(5, 8, 1):
                        cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                                 (0, 255, 0), 3)
                    for t in range(9, 12, 1):
```

```
                cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                         (0, 255, 0), 3)
            for t in range(13, 16, 1):
                cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                         (0, 255, 0), 3)
            for t in range(17, 20, 1):
                cv2.line(white, (self.pts[t][0] + os, self.pts[t][1] + os1), (self.pts[t +
1][0] + os, self.pts[t + 1][1] + os1),
                         (0, 255, 0), 3)
            cv2.line(white, (self.pts[5][0] + os, self.pts[5][1] + os1), (self.pts[9][0] +
os, self.pts[9][1] + os1), (0, 255, 0),
                     3)
            cv2.line(white, (self.pts[9][0] + os, self.pts[9][1] + os1), (self.pts[13][0]
+ os, self.pts[13][1] + os1), (0, 255, 0),
                     3)
            cv2.line(white, (self.pts[13][0] + os, self.pts[13][1] + os1),
(self.pts[17][0] + os, self.pts[17][1] + os1),
                     (0, 255, 0), 3)
            cv2.line(white, (self.pts[0][0] + os, self.pts[0][1] + os1), (self.pts[5][0] +
os, self.pts[5][1] + os1), (0, 255, 0),
                     3)
            cv2.line(white, (self.pts[0][0] + os, self.pts[0][1] + os1), (self.pts[17][0]
+ os, self.pts[17][1] + os1), (0, 255, 0),
                     3)

            for i in range(21):
                cv2.circle(white, (self.pts[i][0] + os, self.pts[i][1] + os1), 2, (0, 0,
255), 1)

            res=white
            self.predict(res)

            self.current_image2 = Image.fromarray(res)

            imgtk = ImageTk.PhotoImage(image=self.current_image2)

            self.panel2.imgtk = imgtk
            self.panel2.config(image=imgtk)

            self.panel3.config(text=self.current_symbol, font=("Courier", 30))
```

```
                #self.panel4.config(text=self.word, font=("Courier", 30))



                self.b1.config(text=self.word1, font=("Courier", 20), wraplength=825,
command=self.action1)
                self.b2.config(text=self.word2, font=("Courier", 20), wraplength=825,
command=self.action2)
                self.b3.config(text=self.word3, font=("Courier", 20), wraplength=825,
command=self.action3)
                self.b4.config(text=self.word4, font=("Courier", 20), wraplength=825,
command=self.action4)

            self.panel5.config(text=self.str, font=("Courier", 30), wraplength=1025)
        except Exception:
            print("==", traceback.format_exc())
        finally:
            self.root.after(1, self.video_loop)

    def distance(self,x,y):
        return math.sqrt(((x[0] - y[0]) ** 2) + ((x[1] - y[1]) ** 2))

    def action1(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str = self.str[:idx_word]
        self.str = self.str + self.word1.upper()


    def action2(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str=self.str[:idx_word]
        self.str=self.str+self.word2.upper()
        #self.str[idx_word:last_idx] = self.word2


    def action3(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
```

```
        self.str = self.str[:idx_word]
        self.str = self.str + self.word3.upper()



    def action4(self):
        idx_space = self.str.rfind(" ")
        idx_word = self.str.find(self.word, idx_space)
        last_idx = len(self.str)
        self.str = self.str[:idx_word]
        self.str = self.str + self.word4.upper()


    def speak_fun(self):
        self.speak_engine.say(self.str)
        self.speak_engine.runAndWait()


    def clear_fun(self):
        self.str=" "
        self.word1 = " "
        self.word2 = " "
        self.word3 = " "
        self.word4 = " "

    def predict(self, test_image):
        white=test_image
        white = white.reshape(1, 400, 400, 3)
        prob = np.array(self.model.predict(white)[0], dtype='float32')
        ch1 = np.argmax(prob, axis=0)
        prob[ch1] = 0
        ch2 = np.argmax(prob, axis=0)
        prob[ch2] = 0
        ch3 = np.argmax(prob, axis=0)
        prob[ch3] = 0

        pl = [ch1, ch2]

        # condition for [Aemnst]
        l = [[5, 2], [5, 3], [3, 5], [3, 6], [3, 0], [3, 2], [6, 4], [6, 1], [6, 2], [6, 6], [6, 7], [6,
0], [6, 5],
             [4, 1], [1, 0], [1, 1], [6, 3], [1, 6], [5, 6], [5, 1], [4, 5], [1, 4], [1, 5], [2, 0], [2,
6], [4, 6],
```

```
            [1, 0], [5, 7], [1, 6], [6, 1], [7, 6], [2, 5], [7, 1], [5, 4], [7, 0], [7, 5], [7, 2]]
        if pl in l:
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                1]):
                ch1 = 0
                # print("00000")

        # condition for [o][s]
        l = [[2, 2], [2, 1]]
        if pl in l:
            if (self.pts[5][0] < self.pts[4][0]):
                ch1 = 0
                print("++++++++++++++++++")
                # print("00000")

        # condition for [c0][aemnst]
        l = [[0, 0], [0, 6], [0, 2], [0, 5], [0, 1], [0, 7], [5, 2], [7, 6], [7, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[4][0] and
self.pts[0][0] > self.pts[12][0] and self.pts[0][0] > self.pts[16][
                0] and self.pts[0][0] > self.pts[20][0]) and self.pts[5][0] > self.pts[4][0]:
                ch1 = 2
                # print("22222")

        # condition for [c0][aemnst]
        l = [[6, 0], [6, 6], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[8], self.pts[16]) < 52:
                ch1 = 2
                # print("22222")



        # condition for [gh][bdfikruvw]
        l = [[1, 4], [1, 5], [1, 6], [1, 3], [1, 0]]
        pl = [ch1, ch2]

        if pl in l:
            if self.pts[6][1] > self.pts[8][1] and self.pts[14][1] < self.pts[16][1] and
self.pts[18][1] < self.pts[20][1] and self.pts[0][0] < self.pts[8][
```

```
            0] and self.pts[0][0] < self.pts[12][0] and self.pts[0][0] < self.pts[16][0] and
self.pts[0][0] < self.pts[20][0]:
                ch1 = 3
                print("33333c")

        # con for [gh][l]
        l = [[4, 6], [4, 1], [4, 5], [4, 3], [4, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][0] > self.pts[0][0]:
                ch1 = 3
                print("33333b")

        # con for [gh][pqz]
        l = [[5, 3], [5, 0], [5, 7], [5, 4], [5, 2], [5, 1], [5, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[2][1] + 15 < self.pts[16][1]:
                ch1 = 3
                print("33333a")

        # con for [l][x]
        l = [[6, 4], [6, 1], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[4], self.pts[11]) > 55:
                ch1 = 4
                # print("44444")

        # con for [l][d]
        l = [[1, 4], [1, 6], [1, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.distance(self.pts[4], self.pts[11]) > 50) and (
                    self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                    self.pts[20][1]):
                ch1 = 4
                # print("44444")

        # con for [l][gh]
        l = [[3, 6], [3, 4]]
        pl = [ch1, ch2]
```

```
        if pl in l:
            if (self.pts[4][0] < self.pts[0][0]):
                ch1 = 4
                # print("44444")

        # con for [l][c0]
        l = [[2, 2], [2, 5], [2, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[1][0] < self.pts[12][0]):
                ch1 = 4
                # print("44444")

        # con for [l][c0]
        l = [[2, 2], [2, 5], [2, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[1][0] < self.pts[12][0]):
                ch1 = 4
                # print("44444")

        # con for [gh][z]
        l = [[3, 6], [3, 5], [3, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
                    1]) and self.pts[4][1] > self.pts[10][1]:
                ch1 = 5
                print("55555b")

        # con for [gh][pq]
        l = [[3, 2], [3, 1], [3, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][1] + 17 > self.pts[8][1] and self.pts[4][1] + 17 > self.pts[12][1]
and self.pts[4][1] + 17 > self.pts[16][1] and self.pts[4][
                    1] + 17 > self.pts[20][1]:
                ch1 = 5
                print("55555a")

        # con for [l][pqz]
        l = [[4, 4], [4, 5], [4, 2], [7, 5], [7, 6], [7, 0]]
```

```
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[4][0] > self.pts[0][0]:
                ch1 = 5
                # print("55555")

        # con for [pqz][aemnst]
        l = [[0, 2], [0, 6], [0, 1], [0, 5], [0, 0], [0, 7], [0, 4], [0, 3], [2, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[0][0] < self.pts[8][0] and self.pts[0][0] < self.pts[12][0] and
self.pts[0][0] < self.pts[16][0] and self.pts[0][0] < self.pts[20][0]:
                ch1 = 5
                # print("55555")

        # con for [pqz][yj]
        l = [[5, 7], [5, 2], [5, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[3][0] < self.pts[0][0]:
                ch1 = 7
                # print("77777")

        # con for [l][yj]
        l = [[4, 6], [4, 2], [4, 4], [4, 1], [4, 5], [4, 7]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[6][1] < self.pts[8][1]:
                ch1 = 7
                # print("77777")

        # con for [x][yj]
        l = [[6, 7], [0, 7], [0, 1], [0, 0], [6, 4], [6, 6], [6, 5], [6, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[18][1] > self.pts[20][1]:
                ch1 = 7
                # print("77777")
        # condition for [x][aemnst]
        l = [[0, 4], [0, 2], [0, 3], [0, 1], [0, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] > self.pts[16][0]:
```

```
            ch1 = 6
            print("666661")

        # condition for [yj][x]
        print("2222  ch1=++++++++++++++++++", ch1, ",", ch2)
        l = [[7, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[18][1] < self.pts[20][1] and self.pts[8][1] < self.pts[10][1]:
                ch1 = 6
                print("666662")

        # condition for [c0][x]
        l = [[2, 1], [2, 2], [2, 6], [2, 7], [2, 0]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[8], self.pts[16]) > 50:
                ch1 = 6
                print("666663")

        # con for [l][x]

        l = [[4, 6], [4, 2], [4, 1], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if self.distance(self.pts[4], self.pts[11]) < 60:
                ch1 = 6
                print("666664")

        # con for [x][d]
        l = [[1, 4], [1, 6], [1, 0], [1, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] - self.pts[4][0] - 15 > 0:
                ch1 = 6
                print("666665")

        # con for [b][pqz]
        l = [[5, 0], [5, 1], [5, 4], [5, 5], [5, 6], [6, 1], [7, 6], [0, 2], [7, 1], [7, 4], [6, 6], [7,
2], [5, 0],
             [6, 3], [6, 4], [7, 5], [7, 2]]
        pl = [ch1, ch2]
        if pl in l:
```

```
            if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
                    1]):
                ch1 = 1
                print("111111")

        # con for [f][pqz]
        l = [[6, 1], [6, 0], [0, 3], [6, 4], [2, 2], [0, 6], [6, 2], [7, 6], [4, 6], [4, 1], [4, 2], [0,
2], [7, 1],
             [7, 4], [6, 6], [7, 2], [7, 5], [7, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and
                    self.pts[18][1] > self.pts[20][1]):
                ch1 = 1
                print("111112")

        l = [[6, 1], [6, 0], [4, 2], [4, 1], [4, 6], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[10][1] > self.pts[12][1] and self.pts[14][1] > self.pts[16][1] and
                    self.pts[18][1] > self.pts[20][1]):
                ch1 = 1
                print("111112")

        # con for [d][pqz]
        fg = 19
        # print("_____________________ch1=",ch1," ch2=",ch2)
        l = [[5, 0], [3, 4], [3, 0], [3, 1], [3, 5], [5, 5], [5, 4], [5, 1], [7, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1]) and (self.pts[2][0] < self.pts[0][0]) and
self.pts[4][1] > self.pts[14][1]):
                ch1 = 1
                print("111113")

        l = [[4, 1], [4, 2], [4, 4]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.distance(self.pts[4], self.pts[11]) < 50) and (
```

```
                self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                self.pts[20][1]):
            ch1 = 1
            print("1111993")

        l = [[3, 4], [3, 0], [3, 1], [3, 5], [3, 6]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1]) and (self.pts[2][0] < self.pts[0][0]) and
self.pts[14][1] < self.pts[4][1]):
                ch1 = 1
                print("1111mmm3")

        l = [[6, 6], [6, 4], [6, 1], [6, 2]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[5][0] - self.pts[4][0] - 15 < 0:
                ch1 = 1
                print("1111140")

        # con for [i][pqz]
        l = [[5, 4], [5, 5], [5, 1], [0, 3], [0, 7], [5, 0], [0, 2], [6, 2], [7, 5], [7, 1], [7, 6], [7,
7]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] > self.pts[20][1])):
                ch1 = 1
                print("111114")

        # con for [yj][bfdi]
        l = [[1, 5], [1, 7], [1, 1], [1, 6], [1, 3], [1, 0]]
        pl = [ch1, ch2]
        if pl in l:
            if (self.pts[4][0] < self.pts[5][0] + 15) and (
            (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
             self.pts[18][1] > self.pts[20][1])):
                ch1 = 7
```

```
            print("111114lll;;p")

        # con for [uvr]
        l = [[5, 5], [5, 0], [5, 4], [5, 1], [4, 6], [4, 1], [7, 6], [3, 0], [3, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if ((self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and
                 self.pts[18][1] < self.pts[20][1])) and self.pts[4][1] > self.pts[14][1]:
                ch1 = 1
                print("111115")

        # con for [w]
        fg = 13
        l = [[3, 5], [3, 0], [3, 6], [5, 1], [4, 1], [2, 0], [5, 0], [5, 5]]
        pl = [ch1, ch2]
        if pl in l:
            if not (self.pts[0][0] + fg < self.pts[8][0] and self.pts[0][0] + fg <
self.pts[12][0] and self.pts[0][0] + fg < self.pts[16][0] and
                    self.pts[0][0] + fg < self.pts[20][0]) and not (
                    self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[12][0] and
self.pts[0][0] > self.pts[16][0] and self.pts[0][0] > self.pts[20][
                0]) and self.distance(self.pts[4], self.pts[11]) < 50:
                ch1 = 1
                print("111116")

        # con for [w]

        l = [[5, 0], [5, 5], [0, 1]]
        pl = [ch1, ch2]
        if pl in l:
            if self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1]:
                ch1 = 1
                print("1117")

        # condn for 8 groups  ends

        # condn for subgroups  starts
        #
        if ch1 == 0:
            ch1 = 'S'
```

```
            if self.pts[4][0] < self.pts[6][0] and self.pts[4][0] < self.pts[10][0] and
self.pts[4][0] < self.pts[14][0] and self.pts[4][0] < self.pts[18][0]:
                ch1 = 'A'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] < self.pts[10][0] and
self.pts[4][0] < self.pts[14][0] and self.pts[4][0] < self.pts[18][
                0] and self.pts[4][1] < self.pts[14][1] and self.pts[4][1] < self.pts[18][1]:
                ch1 = 'T'
            if self.pts[4][1] > self.pts[8][1] and self.pts[4][1] > self.pts[12][1] and
self.pts[4][1] > self.pts[16][1] and self.pts[4][1] > self.pts[20][1]:
                ch1 = 'E'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] > self.pts[10][0] and
self.pts[4][0] > self.pts[14][0] and self.pts[4][1] < self.pts[18][1]:
                ch1 = 'M'
            if self.pts[4][0] > self.pts[6][0] and self.pts[4][0] > self.pts[10][0] and
self.pts[4][1] < self.pts[18][1] and self.pts[4][1] < self.pts[14][1]:
                ch1 = 'N'

        if ch1 == 2:
            if self.distance(self.pts[12], self.pts[4]) > 42:
                ch1 = 'C'
            else:
                ch1 = 'O'

        if ch1 == 3:
            if (self.distance(self.pts[8], self.pts[12])) > 72:
                ch1 = 'G'
            else:
                ch1 = 'H'

        if ch1 == 7:
            if self.distance(self.pts[8], self.pts[4]) > 42:
                ch1 = 'Y'
            else:
                ch1 = 'J'

        if ch1 == 4:
            ch1 = 'L'
        if ch1 == 6:
            ch1 = 'X'

        if ch1 == 5:
            if self.pts[4][0] > self.pts[12][0] and self.pts[4][0] > self.pts[16][0] and
self.pts[4][0] > self.pts[20][0]:
```

```
            if self.pts[8][1] < self.pts[5][1]:
                ch1 = 'Z'
            else:
                ch1 = 'Q'
        else:
            ch1 = 'P'

    if ch1 == 1:
        if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
            1]):
            ch1 = 'B'
        if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
            1]):
            ch1 = 'D'
        if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and self.pts[18][1] > self.pts[20][
            1]):
            ch1 = 'F'
        if (self.pts[6][1] < self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] > self.pts[20][
            1]):
            ch1 = 'I'
        if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] > self.pts[16][1] and self.pts[18][1] < self.pts[20][
            1]):
            ch1 = 'W'
        if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] < self.pts[20][
            1]) and self.pts[4][1] < self.pts[9][1]:
            ch1 = 'K'
        if ((self.distance(self.pts[8], self.pts[12]) - self.distance(self.pts[6],
self.pts[10])) < 8) and (
                self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                self.pts[20][1]):
            ch1 = 'U'
        if ((self.distance(self.pts[8], self.pts[12]) - self.distance(self.pts[6],
self.pts[10])) >= 8) and (
                self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                self.pts[20][1]) and (self.pts[4][1] > self.pts[9][1]):
```

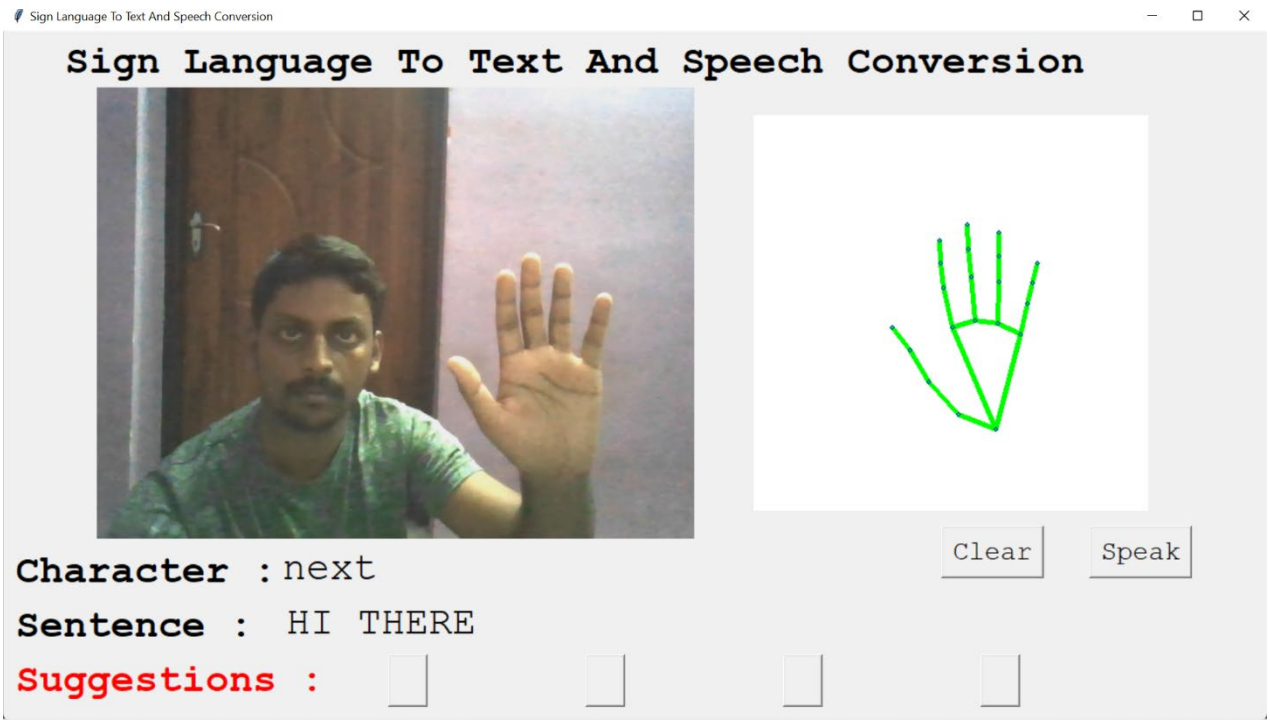
```
            ch1 = 'V'

        if (self.pts[8][0] > self.pts[12][0]) and (
                self.pts[6][1] > self.pts[8][1] and self.pts[10][1] > self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] <
                self.pts[20][1]):
            ch1 = 'R'

    if ch1 == 1 or ch1 =='E' or ch1 =='S' or ch1 =='X' or ch1 =='Y' or ch1 =='B':
        if (self.pts[6][1] > self.pts[8][1] and self.pts[10][1] < self.pts[12][1] and
self.pts[14][1] < self.pts[16][1] and self.pts[18][1] > self.pts[20][1]):
            ch1=" "


    print(self.pts[4][0] < self.pts[5][0])
    if ch1 == 'E' or ch1=='Y' or ch1=='B':
        if (self.pts[4][0] < self.pts[5][0]) and (self.pts[6][1] > self.pts[8][1] and
self.pts[10][1] > self.pts[12][1] and self.pts[14][1] > self.pts[16][1] and
self.pts[18][1] > self.pts[20][1]):
            ch1="next"


    if ch1 == 'Next' or 'B' or 'C' or 'H' or 'F' or 'X':
        if (self.pts[0][0] > self.pts[8][0] and self.pts[0][0] > self.pts[12][0] and
self.pts[0][0] > self.pts[16][0] and self.pts[0][0] > self.pts[20][0]) and (self.pts[4][1]
< self.pts[8][1] and self.pts[4][1] < self.pts[12][1] and self.pts[4][1] < self.pts[16][1]
and self.pts[4][1] < self.pts[20][1]) and (self.pts[4][1] < self.pts[6][1] and
self.pts[4][1] < self.pts[10][1] and self.pts[4][1] < self.pts[14][1] and self.pts[4][1] <
self.pts[18][1]):
            ch1 = 'Backspace'


    if ch1=="next" and self.prev_char!="next":
        if self.ten_prev_char[(self.count-2)%10]!="next":
            if self.ten_prev_char[(self.count-2)%10]=="Backspace":
                self.str=self.str[0:-1]
            else:
                if self.ten_prev_char[(self.count - 2) % 10] != "Backspace":
                    self.str = self.str + self.ten_prev_char[(self.count-2)%10]
        else:
            if self.ten_prev_char[(self.count - 0) % 10] != "Backspace":
                self.str = self.str + self.ten_prev_char[(self.count - 0) % 10]
```

```
        if ch1=="  " and self.prev_char!="  ":
            self.str = self.str + "  "

        self.prev_char=ch1
        self.current_symbol=ch1
        self.count += 1
        self.ten_prev_char[self.count%10]=ch1


        if len(self.str.strip())!=0:
            st=self.str.rfind(" ")
            ed=len(self.str)
            word=self.str[st+1:ed]
            self.word=word
            print("----------word = ",word)
            if len(word.strip())!=0:
                ddd.check(word)
                lenn = len(ddd.suggest(word))
                if lenn >= 4:
                    self.word4 = ddd.suggest(word)[3]

                if lenn >= 3:
                    self.word3 = ddd.suggest(word)[2]

                if lenn >= 2:
                    self.word2 = ddd.suggest(word)[1]

                if lenn >= 1:
                    self.word1 = ddd.suggest(word)[0]
            else:
                self.word1 = " "
                self.word2 = " "
                self.word3 = " "
                self.word4 = " "
    def destructor(self):

        print("Closing Application...")
        print(self.ten_prev_char)
        self.root.destroy()
        self.vs.release()
        cv2.destroyAllWindows()
```

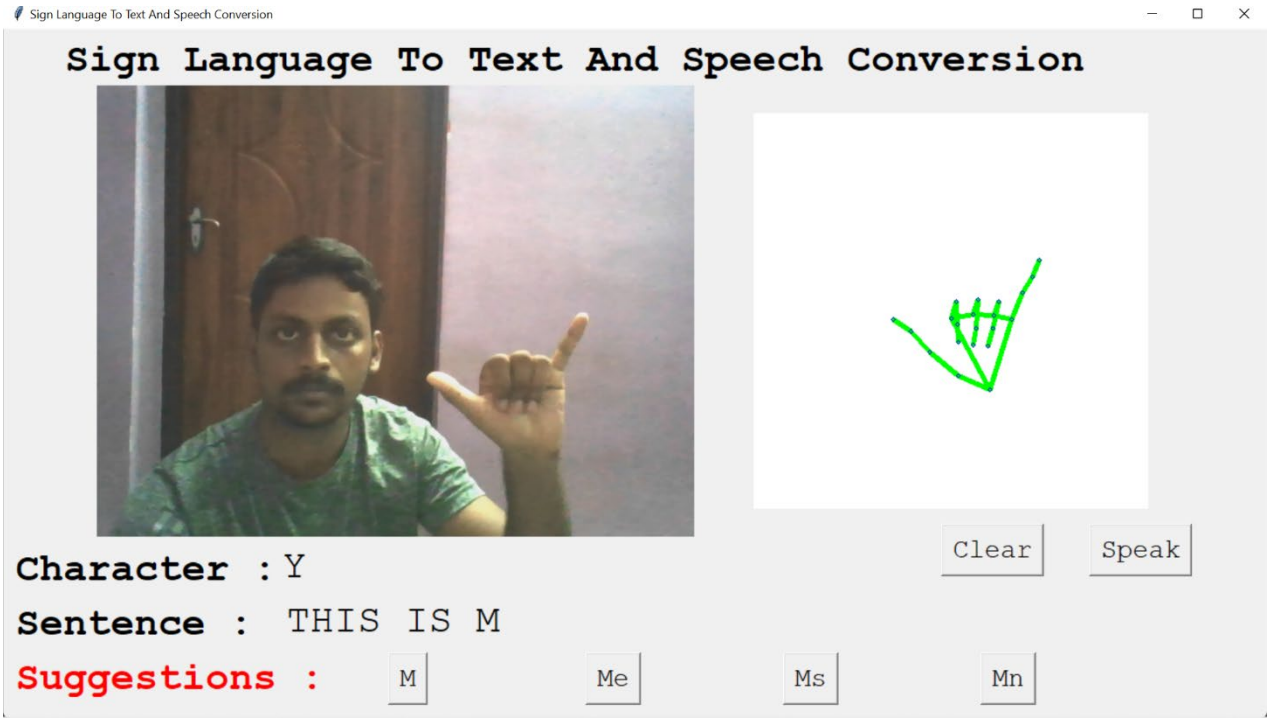
print("Starting Application...")

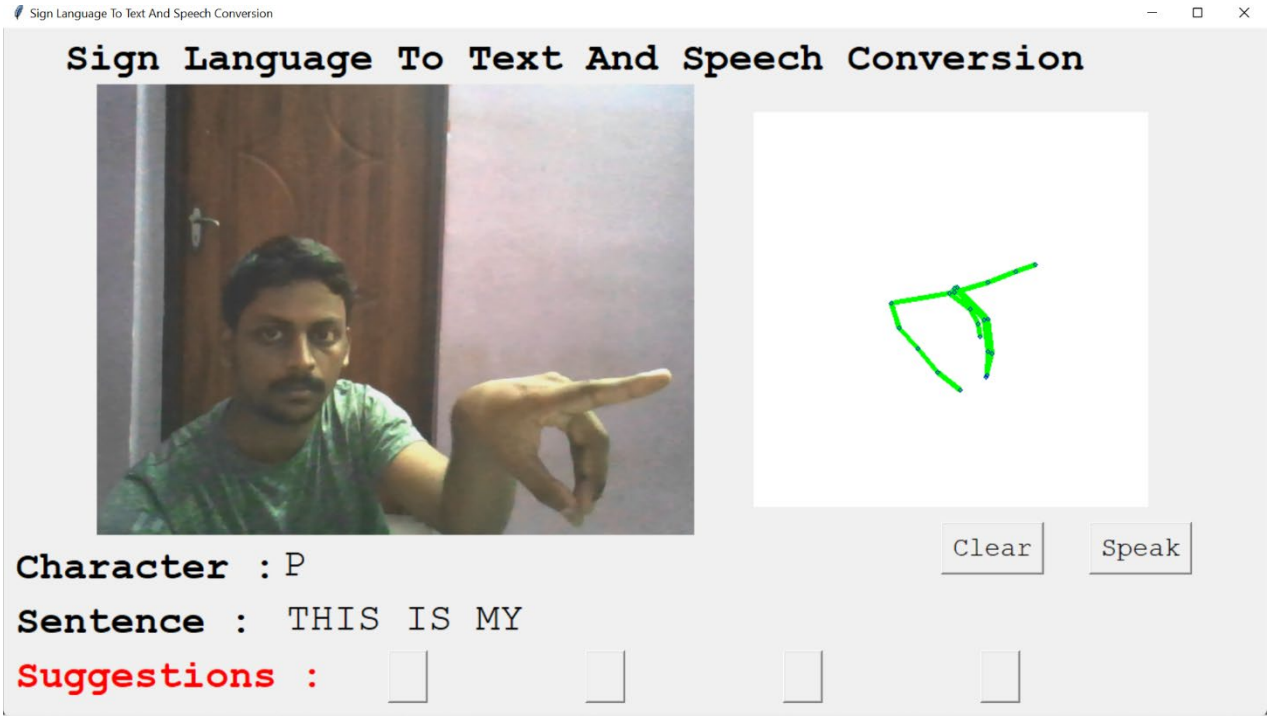
(Application()).root.mainloop()

OUTPUT


Sign Language To Text And Speech Conversion
Sign Language To Text And Speech Conversion
Clear
Speak
Character : next
Sentence : HI THERE
Suggestions :


Sign Language To Text And Speech Conversion
Sign Language To Text And Speech Conversion
Clear
Speak
Character : I
Sentence : THIS
Suggestions :


Sign Language To Text And Speech Conversion
Sign Language To Text And Speech Conversion
Clear
Speak
Character : Y
Sentence : THIS IS M
Suggestions :
M
Me
Ms
Mn


Sign Language To Text And Speech Conversion
Sign Language To Text And Speech Conversion
Clear
Speak
Character : P
Sentence : THIS IS MY
Suggestions :


Sign Language To Text And Speech Conversion
Sign Language To Text And Speech Conversion
Clear
Speak
Character : next
Sentence : THIS IS MY PROJECT
Suggestions :
PROJECTS
PROTECT
PROJECT
PROBE

CHAPTER 9 – CONCLUSION

Many breakthroughs have been made in the field of artificial intelligence, machine learning and computer vision. They have immensely contributed in how we perceive things around us and improve the way in which we apply their techniques in our everyday lives. Many researches have been conducted on sign gesture recognition using different techniques like ANN, LSTM and 3D CNN. However, most of them require extra computing power . On the other hand, our research paper requires low computing power and gives a remarkable accuracy of above 90%.

In our research, we proposed to normalise and rescale our images to 64 pixels in order to extract features (binary pixels) and make the system more robust. We use CNN to classify the 10 alphabetical American sign gestures and successfully achieve an accuracy of 98% which is better than other related work stated in this paper.Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them. The system didn't perform too well but it was demonstrated that it can be built a first-person sign language translation system using only cameras and convolutional neural networks.

It was observed that the model tends to confuse several signs with each other, such as I and J. But thinking a bit about it, maybe it doesn't need to have a perfect performance since using an orthography corrector or a word predictor would increase the translation accuracy. The next step is to analyze the solution and study ways to improve the system. Some improvements could be carrying by collecting more quality data, trying more convolutional neural network architectures, or redesigning the vision system.

We look forward to use more alphabets in our datasets and improve the model so that it recognises more alphabetical features while at the same time get a high accuracy. We would also like to enhance the system by adding speech recognition so that blind people can benefit as well.

CHAPTER 10 – REFERENCES

[1] https://peda.net/id/08f8c4a8511

[2] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle,WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.

[3] CABRERA, MARIA & BOGADO, JUAN & Fermín, Leonardo & Acuña, Raul & RALEV, DIMITAR. (2012). GLOVE-BASED GESTURE RECOGNI-TION SYSTEM. 10.1142/9789814415958_0095.

[4] He, Siming. (2019). Research of a Sign Language Translation System Based on Deep Learning. 392-396. 10.1109/AIAM48774.2019.00083.

[5] International Conference on Trendz in Information Sciences and Computing (TISC). : 30-35, 2012.

[6] Herath, H.C.M. & W.A.L.V.Kumari, & Senevirathne, W.A.P.B & Dissanayake, Maheshi. (2013). IMAGE BASED SIGN LANGUAGE RECOGNITION SYSTEM FOR SINHALA SIGN LANGUAGE

[7] M. Geetha and U. C. Manjusha, , “A Vision Based Recognition of Indian Sign Language Alphabets and Numerals Using B-Spline Approximation”, International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 3, pp. 406-415. 2012.

[8] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L.,

Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. https://doi.org/10.1007/978-3-319-16178-5_40

[9] Escalera, S., Baró, X., Gonzàlez, J., Bautista, M., Madadi, M., Reyes, M., . . . Guyon, I. (2014). ChaLearn Looking at People Challenge 2014: Dataset and Results. Workshop at the European Conference on Computer Vision (pp. 459-473). Springer, . Cham.

[10]Huang, J., Zhou, W., & Li, H. (2015). Sign Language Recognition using 3D convolutional neural networks. IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). Turin: IEEE.

[11]Jaoa Carriera, A. Z. (2018). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 4724-4733). IEEE. Honolulu.

[12]Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE. Miami, FL, USA .

[13]Soomro, K., Zamir , A. R., & Shah, M. (2012). UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. Computer Vision and Pattern Recognition, arXiv:1212.0402v1, 1-7.

[14]Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: a large video database for human motion recognition. Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2556-2563). IEEE

[15]Zhao, Ming & Bu, Jiajun & Chen, C.. (2002). Robust background subtraction in HSV color space. Proceedings of SPIE MSAV, vol. 1. 4861. 10.1117/12.456333.

[16]Chowdhury, A., Sang-jin Cho, & Ui-Pil Chong. (2011). A background subtraction method using color information in the frame averaging process. Proceed- ings of 2011 6th International Forum on Strategic Technology. doi:10.1109/ifost.2011.6021252.