SIGN LANGUAGE RECOGNITION

A MAJOR PROJECT REPORT SUBMITTED TO THE BHARATHIAR UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

Submitted by

SUGESH K

(REG.NO. 22CSEA70)

Under the guidance of

Dr. T. AMUDHA, M.C.A., M.Phil., Ph.D.,

Professor,

Department of Computer Applications

DEPARTMENT OF COMPUTER APPLICATIONS

BHARATHIAR UNIVERSITY

COIMBATORE – 641046

APRIL – 2024

CERTIFICATE

This is to certify that, this major project work entitled "**SIGN LANGUAGE RECOGNITION"** was submitted to the Department of Computer Applications, Bharathiar University in partial fulfillment of the requirements for the award of the degree of **MASTER OF COMPUTER APPLICATIONS**, is a record of original work done by **SUGESH K (22CSEA70),** during his period of study in the Department of Computer Applications, Bharathiar University, Coimbatore, under my supervision and guidance, and this project work has not formed the basis for the award of any Degree/ Diploma /Associateship/ Fellowship or similar title to any candidate of any University.

Place : Coimbatore

Date :

Submitted for the University Viva-Voce Examination held on _____

Project Guide **Head of the Department**

Internal Examiner **External Examiner**

DECLARATION

I hereby declare that this major project work titled, "**SIGN LANGUAGE RECOGNITION**" submitted to Department of Computer Applications, Bharathiar University, is a record of original work done by **SUGESH K(22CSEA70)**, under the supervision and guidance of **Dr. T. AMUDHA, M.C.A., M.Phil., Ph.D.,** Professor Department of Computer Applications, Bharathiar University, and that this project work has not formed the basis for theaward of any Degree/ Diploma/ Associateship/Fellowship or similar title to any candidate of any University.

Place: Coimbatore **Signature of the candidate**

Date: **(SUGESH K)**

Countersigned by

Dr. T. AMUDHA

Professor,

Department of Computer Applications

Bharathiar University

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I express my sincere gratitude to our Professor& Head of the Department **Dr. M. PUNITHAVALLI, M.Sc., M.Phil., Ph.D.,** Department of Computer Application, Bharathiar University, Coimbatore, thanks to department faculty members and guest faculties supported to acquire knowledge from various inputs. I am forever indebted to this mini project guides **Dr. T. AMUDHA, M.C.A., M.Phil., Ph.D.,** Department of Computer Applications, Bharathiar University for providing valuable suggestions and always been enthusiastically guiding us throughout the project.

Finally, I also extend my special thanks to my family, friends, who havekindly provided the necessary support for the successful completion of the project andtheir moral support.

SYNOPSIS

Sign language is one of the oldest and most natural form of language for communication, but since most people do not know sign language and interpreters are very difficult to come by, to have come up with a real time method using neural networks for fingerspelling based American sign language. In this method, the hand is first passed through a filter and after the filter is applied the hand is passed through a classifier which predicts the class of the hand gestures. In this method provides 95.7 % accuracy for the 26 letters of the alphabet.

Traditional Sign language recognition systems have primarily relied on computer vision algorithms to interpret gestures captured through video feeds. However, recent developments in deep learning have revolutionized the field by enabling more nuanced understanding of sign language expressions. Convolutional Neural Networks (CNNs) have shown remarkable capabilities in extracting spatial features from sign language images, while Recurrent Neural Networks (RNNs) and Transformers have been effective in modeling temporal dependencies in sign sequences.

The utilization of transfer learning techniques, where pre-trained models are fine-tuned on sign language datasets, has expedited the development of SLR systems, especially in scenarios with limited labeled data the integration of advanced machine learning techniques holds great promise for advancing the field of sign language recognition, ultimately fostering inclusivity and accessibility for individuals with hearing impairments in various domains of daily life.

CHAPTER 1
INTRODUCTION

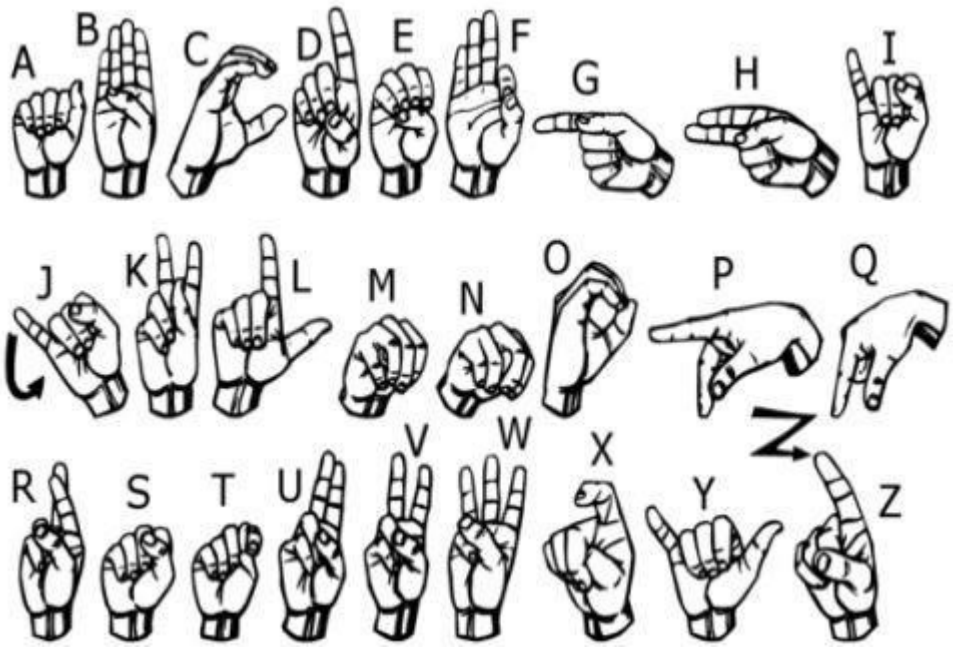
1.1 Sign Language Recognition in Python

American sign language is a predominant sign language Since the only disability Deaf and Dumb (hereby referred to as D&M) people have is communication related and since they cannot use spoken languages, 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. D&M people make use of their hands to express different gestures to express their ideas with other people. Gestures are the non-verbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

A sign language is a language which uses gestures instead of sound to convey meaning combining hand-shapes, orientation and movement of the hands, arms or body, facial expressions and lip-patterns. Contrary to popular belief, sign language is not international. These vary from region to region.

Minimizing the verbal exchange gap among D&M and non-D&M people turns into a want to make certain effective conversation among all. Sign language translation is among one of the most growing lines of research and it enables the maximum natural manner of communication for those with hearing impairments. A hand gesture recognition system offers an opportunity for deaf people to talk with vocal humans without the need of an interpreter. The system is built for the automated conversion of ASL into textual content and speech.

In this project we primarily focus on producing a model which can recognize 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 Trained Symbols

CHAPTER 2
LITERATURE REVIEW

Early Research (Pre-2000s):

Before the 2000s, research on sign language recognition was limited due to technological constraints. Early studies often focused on manual coding of sign language or basic gesture recognition systems.

Emergence of Computer Vision (2000s):

With advancements in computer vision and machine learning, researchers started exploring automated sign language recognition. Techniques like Hidden Markov Models (HMMs) and Neural Networks began to be applied.

Gesture Recognition and Machine Learning (2010s):

The 2010s saw significant progress in sign language recognition, primarily driven by advances in deep learning. Convolutional Neural Networks (CNNs) became popular for extracting features from video data.

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, were employed for sequential modeling of sign language gestures. Research focused on improving accuracy, robustness to variations in signing styles, and real-time performance.

Datasets and Benchmarks:

Development of large-scale sign language datasets became crucial for training and evaluating models. Datasets like RWTH-PHOENIX-Weather 2014T and American Sign Language Lexicon Video Dataset (ASLLVD) provided valuable resources for researchers. Benchmarks such as the Sign Language Recognition and Translation Challenge (SLRT) facilitated comparison of different methods.

Challenges and Solutions:

Challenges in sign language recognition include dealing with variations in signing speed, occlusions, and background clutter. Researchers proposed solutions such as data augmentation techniques, attention mechanisms, and multimodal fusion (combining video and skeletal data).Domain adaptation techniques were explored to address domain shifts between lab-controlled datasets and real-world scenarios.

Real-time Applications and Accessibility:

Efforts were made to develop real-time sign language recognition systems for practical applications. Applications included sign language translation, assistive technologies for the deaf and hard of hearing, and human-computer interaction.

Recent Advances (Post-2020):

Recent research has focused on improving the interpretability and explain ability of sign language recognition models. Techniques such as attention mechanisms, Transformer architectures, and self-supervised learning have shown promising results. There's a growing emphasis on addressing biases in datasets and ensuring inclusivity in sign language technologies.

Frameworks and Tools:

Python libraries like OpenCV, TensorFlow, and PyTorch have been instrumental in implementing sign language recognition systems. Open-source projects and frameworks specifically designed for sign language recognition, such as Sign Language Transformer (SLT), have emerged.

CHAPTER 3

METHODOLOGIES

3.1 METHODS

The system is a vision-based approach. All signs are represented with bare hands and so it eliminates the problem of using any artificial devices for interaction.

3.1.1 Data set generation

For the project we tried to find already made datasets but we couldn't find dataset in the form of raw images that matched our requirements. All we could find were the datasets in the form of RGB values. Hence, we decided to create our own data set. Steps we followed to create our data set are as follows.

It used Open computer vision (OpenCV) library in order to produce our dataset.

Firstly, we captured around 800 images of each of the symbol in ASL (American Sign Language) for training purposes and around 200 images per symbol for testing purpose.

3.1.2. Gesture Classification

Our approach uses two layers of algorithm to predict the final symbol of the user.

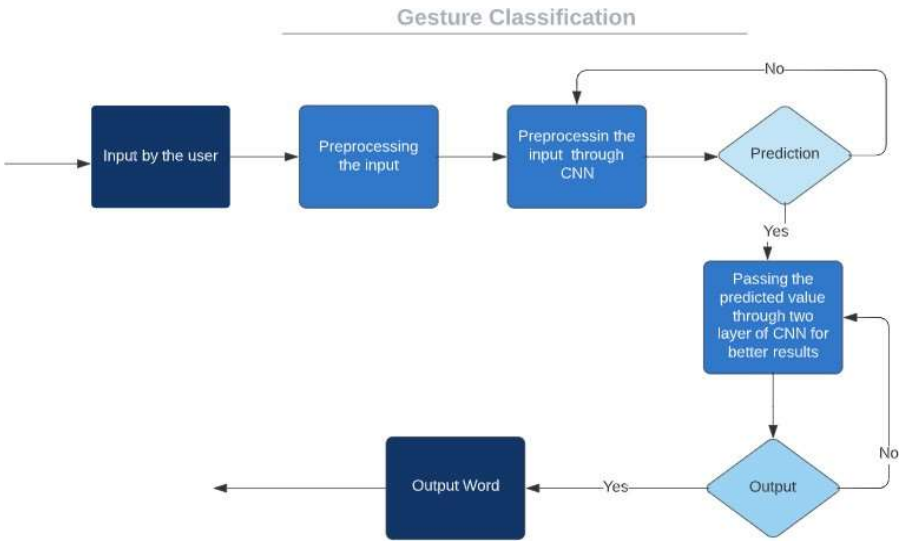


Fig-3.1 Gesture classification

Algorithm Layer 1:

1. Apply Gaussian Blur filter and threshold to the frame taken with openCV to get the processed image after feature extraction.
2. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word.
3. Space between the words is considered using the blank symbol.

Algorithm Layer 2:

1. It detect various sets of symbols which show similar results on getting detected.
2. Then classify between those sets using classifiers made for those sets only.

Layer 1:

CNN Model:

1. **1st Convolution Layer:** The input picture has resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weights.
2. **1st Pooling Layer:** The pictures are down sampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.
3. **2nd Convolution Layer:** Now, these 63 x 63 from the output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 60 x 60 pixel image.
4. **2nd Pooling Layer:** The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.
5. **1st Densely Connected Layer:** Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.

6. **2nd Densely Connected Layer:** Now the output from the 1st Densely Connected Layer is used as an input to a fully connected layer with 96 neurons.
7. **Final layer:** The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

Activation Function:

It have used ReLU (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons).

ReLU calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

Pooling Layer:

Apply the **Max** pooling to the input image with a pool size of (2, 2) with ReLU activation function. This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

Dropout Layers:

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. This layer "drops out" a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out [5].

Optimizer:

Adam optimizer for updating the model in response to the output of theloss function.

Adam optimizer combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp).

Layer 2:

The two layers of algorithms to verify and predict symbols which are more similar to each other so that we can get us close as we can get to detect the symbol shown. In our testingwe found that following symbols were not showing properly and were giving other symbols also:

1. **For D : R and U**
2. **For U : D and R**
3. **For I : T, D, K and I**
4. **For S : M and N**

So, to handle above cases we made three different classifiers for classifying these sets:

1. **{D, R, U}**
2. **{T, K, D, I}**
3. **{S, M, N}**

3.1.3 Finger Spelling Sentence Formation Implementation

1. Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold, we print the letter and add it to the current string (In our code we kept the value as 50 and difference threshold as 20).
2. Otherwise, we clear the current dictionary which has the count of detections of present symbol to avoid the probability of a wrong letter getting predicted.
3. Whenever the count of a blank (plain background) detected exceeds a specific value and if the current buffer is empty no spaces are detected.
4. In other case it predicts the end of word by printing a space and the current gets appended to the sentence below.

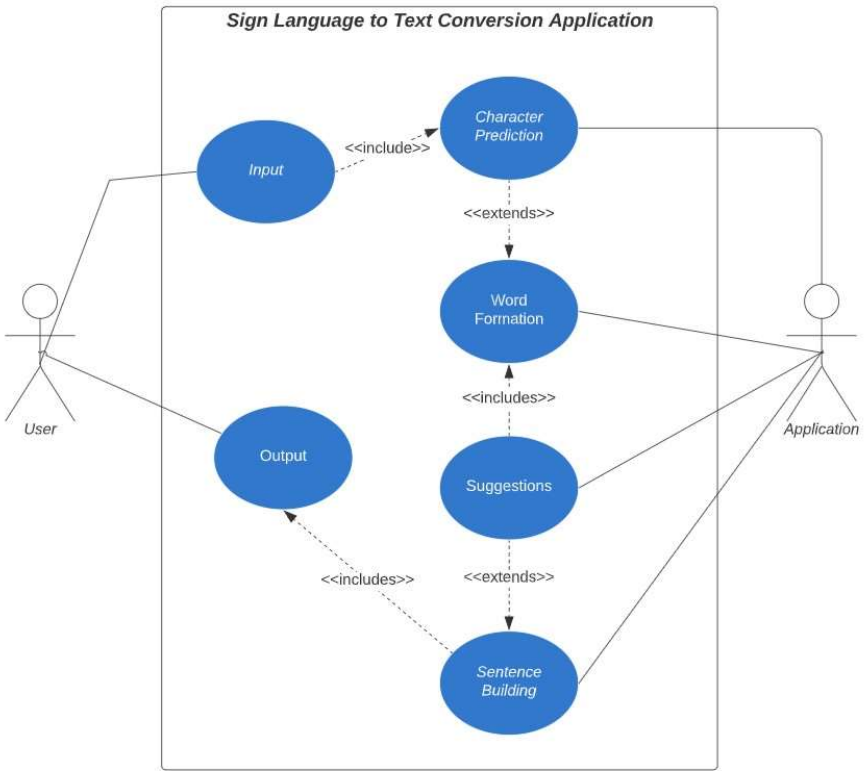


Fig – 3.2 Flow Diagram for sign language to text conversion

AutoCorrect Feature:

A python library Hunspell suggest is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence. This helps in reducing mistakes committed in spellings and assists in predicting complex words.

3.1.4 Training and testing

There convert our input images (RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 128 x 128.

It feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above.

The prediction layer estimates how likely the image will fall under one of the classes. So, the output is normalized between 0 and 1 and such that the sum of each value in each class sums to 1. We have achieved this using SoftMax function.

At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labelled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labelled value and is zero exactly when it is equal to the labelled value. Therefore, we optimized the cross-entropy by minimizing it as close to zero. To do this in our network layer we adjust the weights of our neural networks. TensorFlow has an inbuilt function to calculate the cross entropy.

As there found out the cross-entropy function, we have optimized it using Gradient Descentin fact with the best gradient descent optimizer is called Adam Optimizer.

3.2 PACKAGES

NumPy:

NumPy is often used in sign language recognition because of its efficiency in handling numerical operations and multidimensional arrays, which are essential for processing image and video data. Sign language recognition involves analyzing video streams or image sequences of sign language gestures, converting them into numerical representations, and then applying various algorithms for classification or recognition. NumPy integrates well with other libraries commonly used in machine learning and computer vision, such as OpenCV and sci-kit-learn. This allows for seamless integration of various components in the sign language recognition pipeline.

Strings:

In sign language recognition, strings are often used to represent the linguistic content of the signs. While sign language is primarily a visual language, the linguistic content of signs can be essential for accurate interpretation and translation. Overall, using strings in sign language recognition enables interoperability with text-based systems, supports language processing tasks, facilitates translation, assists in data annotation, and enhances accessibility for a wider audience.

OS, SYS:

Sign language recognition systems often deal with a large amount of data, including video files, image sequences, and trained models. The os module provides functions for interacting with the file system, such as creating, moving, renaming, and deleting files and directories. This is crucial for organizing and managing datasets, model files, and other resources used in the recognition system.

OpenCV

OpenCV (Open-Source Computer Vision Library) is a widely used open-source computer vision and image processing library. OpenCV provides extensive functionalities for reading, manipulating, and processing images and video streams. In sign language recognition, input data often comes from video feeds or image sequences, and OpenCV enables tasks such as frame extraction, resizing, filtering, and color space conversion.

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). Keras provides access to a wide range of pre-trained models through its keras.applications module. These pre-trained models, trained on large-scale datasets like ImageNet, can be fine-tuned for sign language recognition tasks, saving time and computational resources compared to training models from scratch.

TKINTER

Tkinter is primarily a GUI toolkit for building desktop applications with Python, it may not be directly involved in the core functionality of sign language recognition systems. Tkinter itself may not directly handle the sign language recognition algorithms, it can be used to create the user interface, manage user interactions, and facilitate integration with other components, thereby enhancing the usability and functionality of sign language recognition applications.

Pillow:

The Pillow module (Python Imaging Library, or PIL) is a powerful library for opening, manipulating, and saving many different image file formats. In sign language recognition, Pillow can be utilized for various image processing tasks, which are crucial for preprocessing input images before feeding them into the recognition system.

Hunspell

The Hunspell module is primarily used for spell checking and morphological analysis, particularly for text written in natural languages. In the context of sign language recognition, where the primary input is visual (video or images of sign language gestures), the Hunspell module might not have a direct application in the core recognition process.

CHAPTER 4

CHALLENGES FACED

There were many challenges faced during the project. The very first issue we faced was that concerning the data set. We wanted to deal with raw images and that too square images as CNN in Keras since it is much more convenient working with only square images.

It couldn't find any existing data set as per our requirements and hence we decided to make our own data set. Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provide that image as input for CNN model.

As tried various filters including binary threshold, canny edge detection, Gaussian blur etc. but finally settled with Gaussian Blur Filter.

More issues were faced relating to the accuracy of the model we had trained in the earlier phases. This problem was eventually improved by increasing the input image size and also by improving the data set.

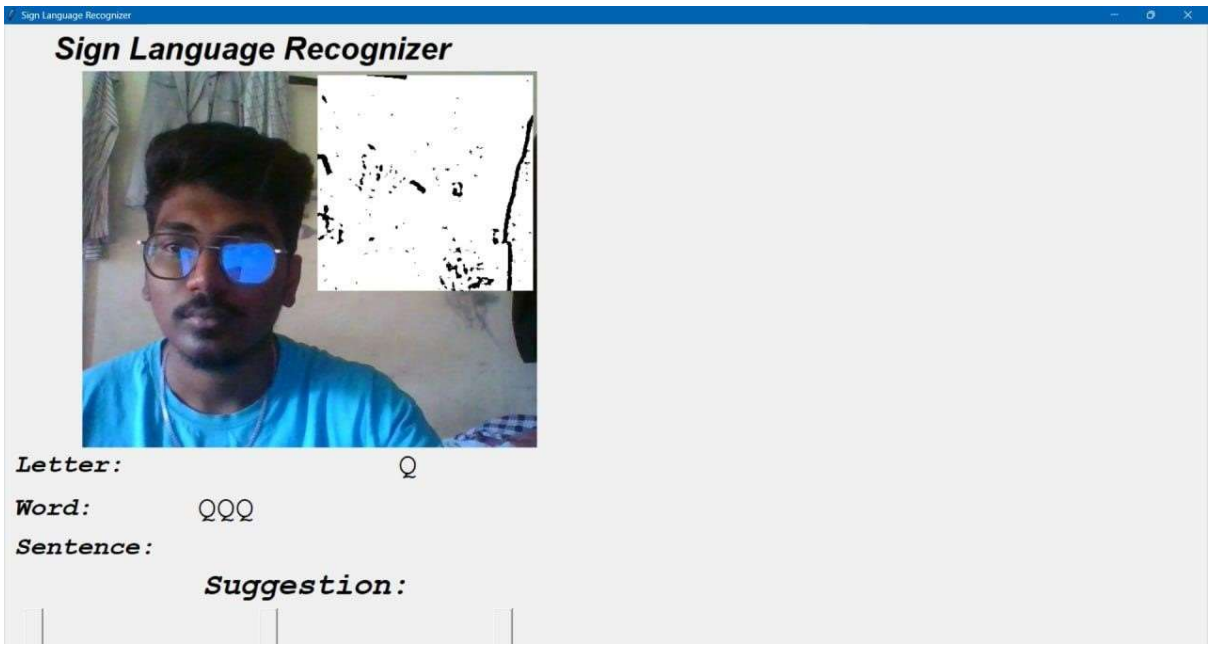


Fig 4.1 Disturbance in Frame

CHAPTER 5

OUTPUT AND RESULT INTERPRETATION

Input:

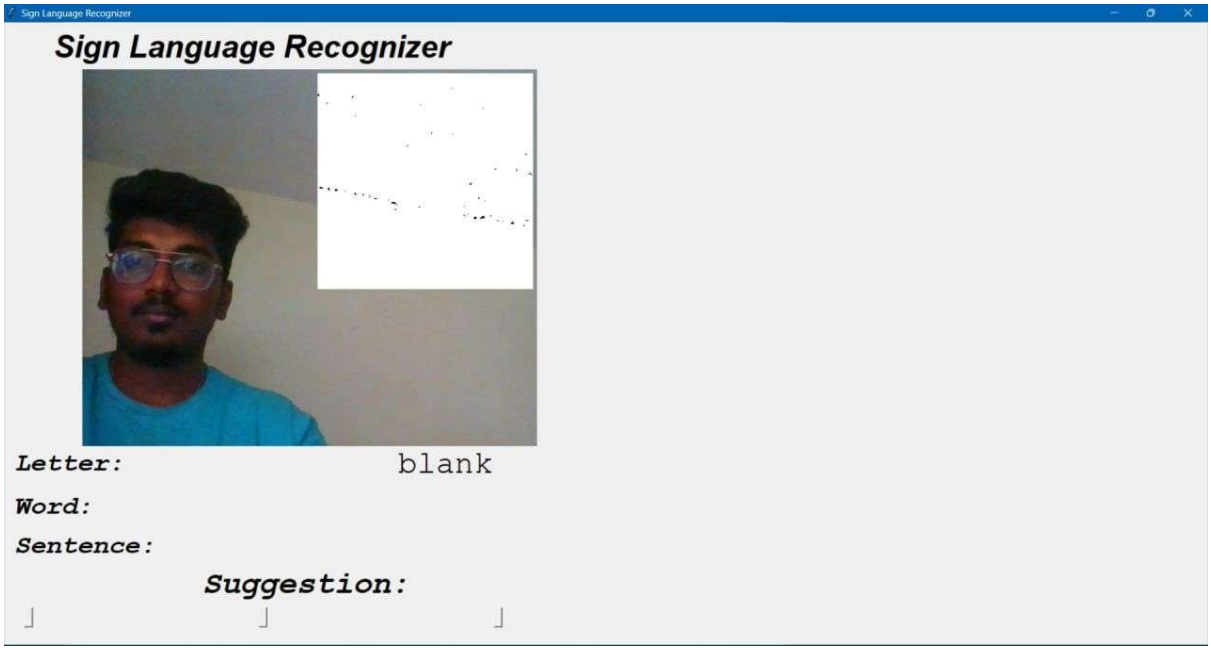


Fig 5.1 First Page

In this input design, it contains a letter area, word area, and sentence area and also have in three suggestion box. First in letter area the suggested letter will be shown and the suggested letter will be accurate it will be shown in word area.

Recognize The Symbol:

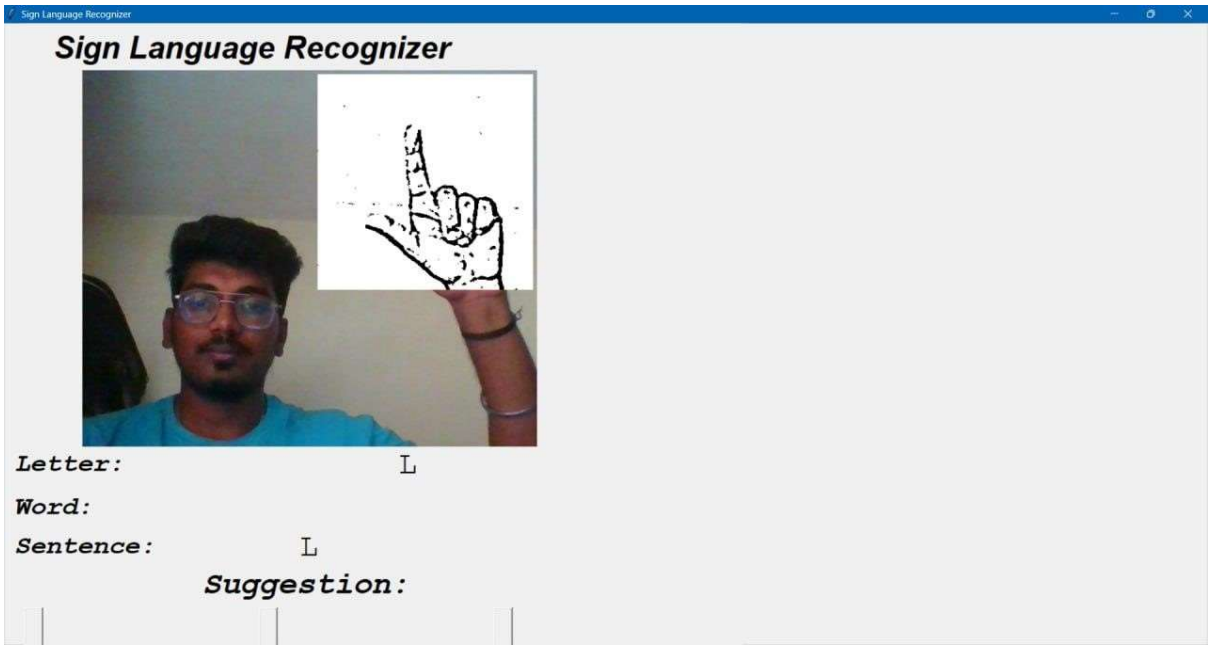


Fig 5.2 Recognize the Symbols

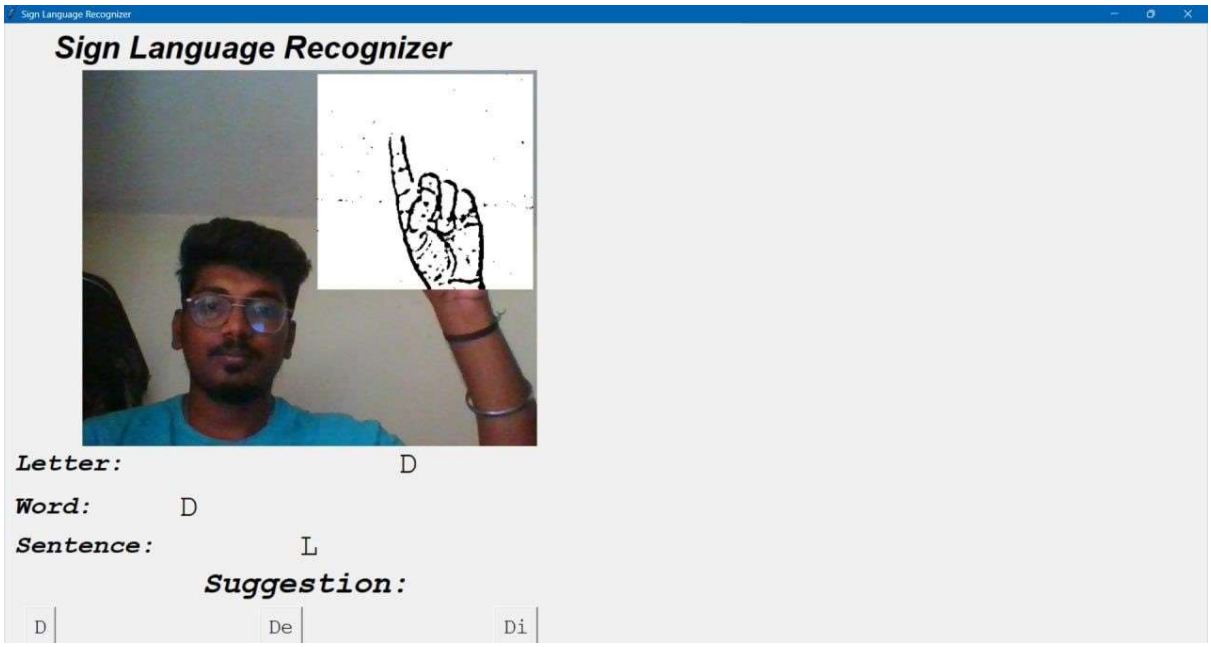


Fig 5.3 Conform the Symbol

Once the letter is typed in the word area then symbol has accurate value

Building a Word

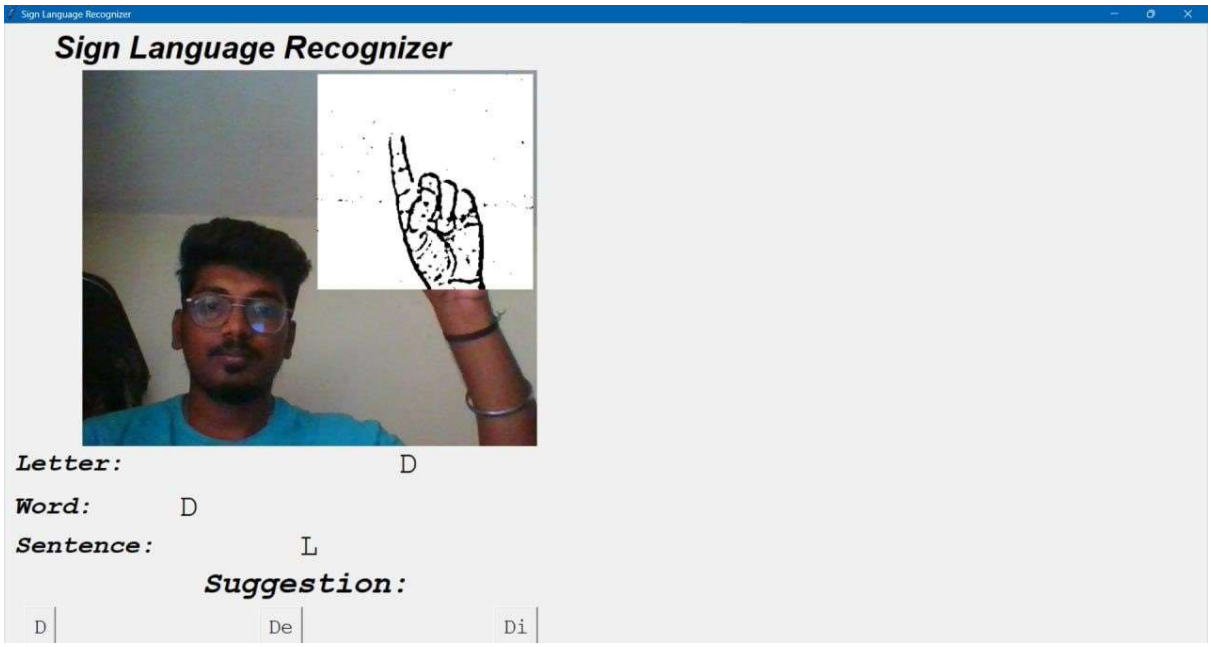


Fig 5.4 Forming the first Word

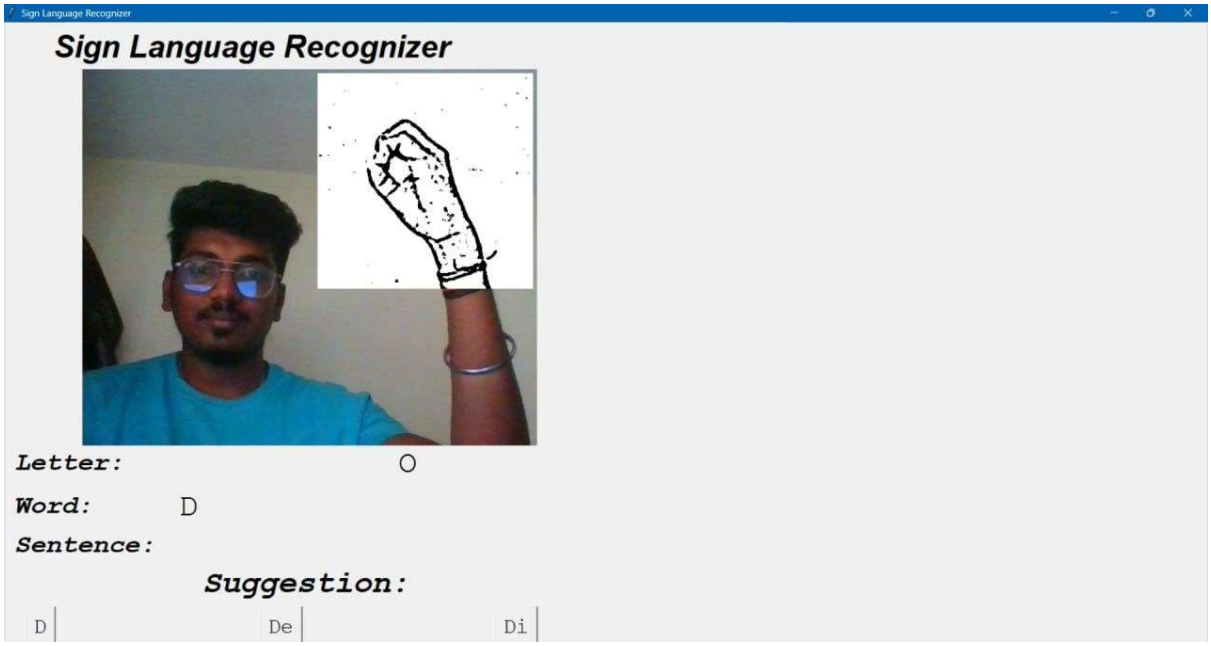


Fig 5.5 Forming the second Word

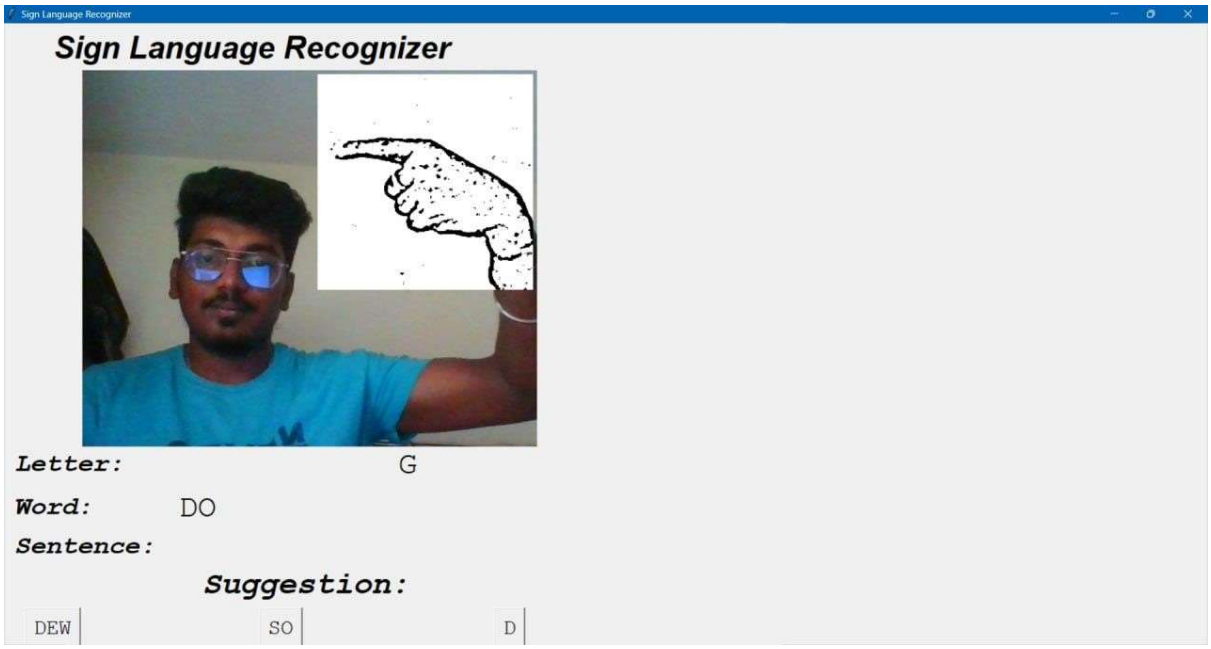


Fig 5.6 Forming a third Word

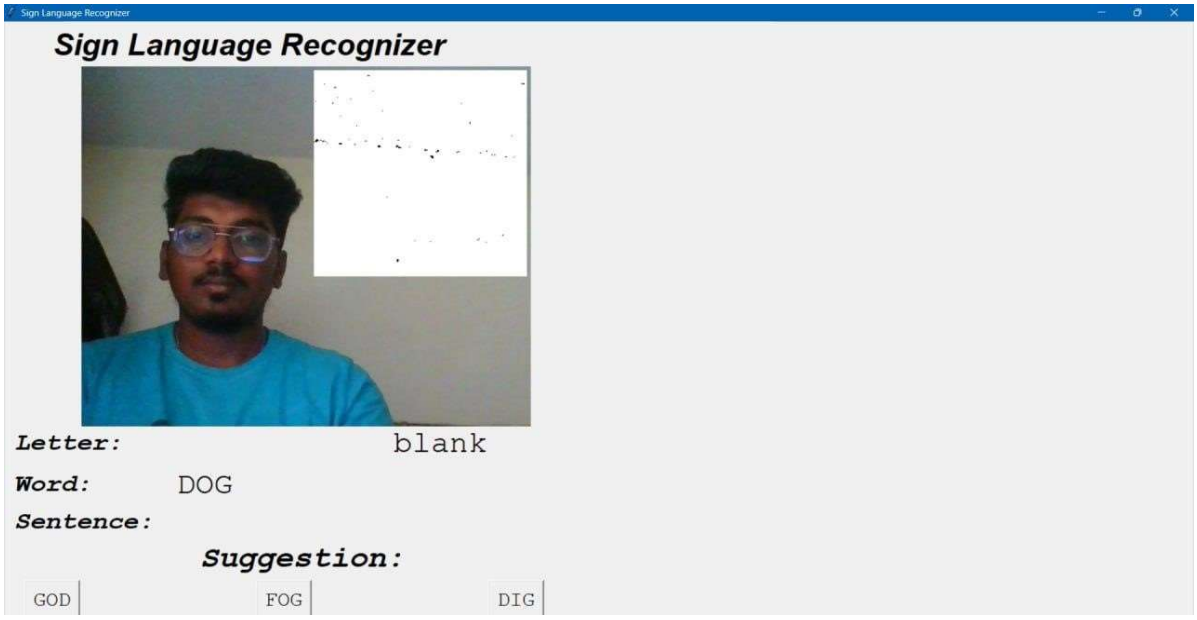


Fig 5.7 list the suggestion words

Once something has formed or recognized in the word area there are some three different words which is similar to word area that has been showed in suggestion area

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

Sign language is a complex visual language with intricate gestures and expressions. Recognizing and interpreting sign language gestures accurately requires sophisticated algorithms capable of capturing subtle nuances in movement and form.

The achieved final accuracy of 90.0% on our data set. We have improved our prediction afterimplementing two layers of algorithms wherein we have verified and predicted symbols which are more similar to each other. This gives us the ability to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate. Building effective sign language recognition systems starts with collecting high-quality data sets of sign language gestures. This involves recording diverse signers performing a wide range of gestures in various environments.

User interfaces for sign language recognition systems should be intuitive, responsive, and accessible to users with varying levels of sign language proficiency. Incorporating feedback mechanisms, error handling, and customization options can enhance the user experience and improve system usability. sign language recognition projects require interdisciplinary collaboration, careful design, and rigorous evaluation to develop effective and inclusive systems that empower sign language users and promote communication accessibility.

6.2 FUTURE ENHANCEMENT

This project can be enhanced by being built as a web/mobile application for the users to conveniently access the project. Also, the existing project only works for ASL; it can be extended to work for other native sign languages with the right amount of data set and training. This project implements a finger spelling translator; however, sign languages are also spoken in a contextual basis where each gesture could represent an object, or verb. So, identifying this kind of a contextual signing would require a higher degree of processing and natural language processing (NLP).

BIBLIOGRAPHY

1. Byeongkeun Kang, Subarna Tripathi, Truong Q. Nguyen" Real-time sign language fingerspelling recognition using convolutional neural networks from depth map" 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)

2. Mohammed Waleed Kalous, Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language.

3. N. Mukai, N. Harada and Y. Chang, "Japanese Fingerspelling Recognition Based on Classification Tree and Machine Learning," *2017 Nicograph International (NicoInt)*, Kyoto, Japan, 2017, pp. 19-24. doi:10.1109/NICOInt.2017.9

4. 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

5. T. Yang, Y. Xu, and "A., Hidden Markov Model for Gesture Recognition", CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA, May 1994.

WEBSITES

1.https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

2. http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php

3. https://en.wikipedia.org/wiki/TensorFlow

4. https://en.wikipedia.org/wiki/Convolutional_neural_nework

5. https://opencv.org/