SIGN LANGUAGE RECOGNITION SYSTEM

A PROJECT REPORT

Submitted by

BHAVYA SHARMA	**(20BCS7867)**	Bavya
RAGINI PANDEY	**(20BCS7891)**	Ragini
VIDHI THAPA	**(20BCS1301)**	Vidhi
P. RAMESH JOGARAO	**(20BCS7868)**	Rajesh
MRITUNJAY PANDEY	**(20BCS7851)**	Mritunjay

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

OCTOBER 2022

BONAFIDE CERTIFICATE

Certified that this project report "SIGN LANGUAGE RECOGNITION SYSTEM" is the bonafide work of "BHAVYA SHARMA, RAGINI PANDEY, VIDHI THAPA, POTHANAPALLI RAMESH JOGARAO, MRITUNJAY PANDEY" who carried out the project work under my/our supervision.

SIGNATURE **SIGNATURE**

HEAD OF THE DEPARTMENT **SUPERVISOR**

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our profound gratitude to **Dr. Sandeep Kang** (HOD), of Computer Science and Engineering department, and **Mr. B. Priestly Shaan** (Dean) of Chandigarh university for their contributions to the completion of my project titled Sign Language Recognition.

We would like to express our special thanks to our supervisor **Er. Kushwant Kaur (E11447)** and co-supervisor **Dr. Himanshu Sharma (E13027)** for their time and efforts they provided throughout the year. Your useful advice and suggestions were really helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

We would like to acknowledge that this project was completed entirely by us .

Signature

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Sign Language is mainly used by deaf (hard hearing) and dumb people to exchange information between their own community and with other people. It is a language where people use their hand gestures to communicate with others as they can't speak or hear. Not everyone possesses the knowledge and understanding of sign language which makes communication difficult between a normal person and a deaf and dumb person. To overcome this barrier , we come up with a model based on machine learning. This model is trained to recognize different gestures of sign language and translate them into English. This will help a lot of people in communicating with deaf and dumb people. Our Sign Language Recognition system works on an algorithm for recognition of hand gestures with accuracy. In this machine learning project , we are going to make a real-time Sign Language Recognize with the help of OpenCV , CNN algorithm , keras and tensorflow . The main goal of this project is to create a computer-based intelligent system that will allow deaf persons to interact effectively with others by using hand gestures.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Sign language[1] is the mode of communication which uses visual ways like expressions, hand gestures[2], and body movements to convey meaning. Sign language[1] is extremely helpful for people who face difficulty with hearing or speaking. Sign language recognition[1] refers to the conversion of these gestures into words or alphabets of existing formally spoken languages. Thus, conversion of sign language into words by an algorithm or a model can help bridge the gap between people with hearing or speaking impairment and the rest of the world.

Vision-based hand gesture recognition[3] is an area of active current research in computer vision and machine learning. Being a natural way of human interaction, it is an area where many researchers are working on, with the goal of making human computer interaction [4](HCI) easier and natural, without the need for any extra devices. So, the primary goal of gesture recognition research is to create systems, which can identify specific human gestures and use them, for example, to convey information. For that, vision-based hand gesture[2] interfaces require fast and extremely robust hand detection, and gesture recognition in real time. Hand gestures[2] are a powerful human communication modality with lots of potential applications and in this context, we have sign language recognition, the communication method of deaf people.

Hand gesture recognition[5] for human computer interaction[6] is an area of active research in computer vision and machine learning. One of its primary goals is to create systems, which can identify specific gestures and use them to convey information or to control a device. Though, gestures need to be modeled in the spatial and temporal domains, where a hand posture is the static structure of the hand and a gesture is the dynamic movement of the hand. There are basically two types of approaches for hand

gesture recognition[2]: vision-based approaches and data glove approaches. This main focus is on creating a vision-based system able to do a real-time sign language recognition. The reason for choosing a system based on vision relates to the fact that it provides a simpler and more intuitive way of communication between a human and a computer. Being hand-pose one of the most important communication tools in humans daily life, and with the continuous advances of image and video processing techniques, research on human-machine interaction[7] through gesture recognition led to the use of such technology in a very broad range of applications, like touch screens, video game consoles, virtual reality, medical applications, and sign language recognition. Although sign language is the most natural way of exchanging information among deaf people it has been observed that they are facing difficulties with normal people interaction. Sign language consists of vocabulary of signs in exactly the same way as spoken language consists of a vocabulary of words. Sign languages are not standard and universal and the grammars differ from country to country.

1.2 Objectives

1. To recognise the meaning of the hand gestures and showing it as output.
2. To help deaf and dumb people to communicate without much problem.
3. To suggest word predictions to the already provided output
4. To recognise the hand gestures with good accuracy.
5. To make an easy user interface to be able to have the input and output in the same window,

1.3 Problem Identification

Sign language recognition[8] is a problem that has been addressed in research for years. However, we are still far from finding a complete solution available in our

society. Sign language uses lots of gestures so that it looks like a movement language which consists of a series of hands and arms motions. For different countries, there are different sign languages and hand gestures. Also, it is noted that some unknown words are translated by simply showing gestures for each alphabet in the word. In addition, sign language also includes specific gestures to each alphabet in the English dictionary and for each number between 0 and 9. This work focuses on the creation of a real time software system that will be able to recognize hand-gestures using deep learning techniques[3]. This project aims to predict the 'alphanumeric' gesture of the sign language system[9].

1.4 Planning

1.4.1 Data Collection

In this Phase we will collect the images of different hand gestures using a webcam. We will have different folders for different gestures. Approximately 100-200 images per gesture.This is the fundamental and important phase of our project.

1.4.2 Image Processing

In this phase we will be processing the images that we collected in the previous step. Processing of images is necessary in this project to remove unwanted noise and unwanted objects. We will apply threshold and gaussian blur to make our hand boundaries more defined, which will help us in analyzing the images more accurately.

1.4.3 Training

In this step we will be training our program to analyze the Images, so that we can have accurate predictions afterwards. Training is the most important phase of a machine learning project.

1.4.4 Predictions

In this step we will be extracting predictions on the basis of what is given as input. If it is predicting the right result then it will be confirmed, otherwise we have to train it further to minimize the error.

1.4.5 GUI Development

In this step we will be developing the User interface through which we will be giving input and getting results in the same window. The user interface must be simple and easy to use.

1.4.6 Testing

In this step we will be testing the outputs of the Gestures. If the program is predicting the result right, it will be confirmed. If not then we have to train it further to reduce the risk of error.

CHAPTER 2
LITERATURE SURVEY

The identification of hand gestures has been the subject of extensive research in recent years. With the help of literature

survey done we realized the basic steps in hand gesture recognition are :

Data acquisition:

There are various methods for gathering information on hand gestures in the subsequent ways:

For precise hand positioning, electromechanical devices are used. position and arrangement. Various glove-based techniques can be an information extraction tool. But it is pricey and difficult to use.

Computer cameras are used as the input device for hands or finger information in vision-based approaches. The Vision Based approaches just need a camera, enabling a seamless contact between people and computers without the need for any additional hardware.

These devices often augment biological eyesight by systems that use artificial vision that are implemented in hardware or software.

The fundamental difficulty in vision-based hand detection[10] is coping with the considerable variability in the look of human hands caused by a great number of hand movements, a variety of skin color options, as well as differences in view points, scales, and camera speeds used to capture the image.

Data Preprocessing

The method for hand detection in [11] incorporates background subtraction and threshold-based color detection[12]. Because hands and faces both entail comparable skin tones, we can distinguish between them using the Adaboost face detector.

By using a filter known as Gaussian blur, we may also extract the necessary image that will be used for training. OpenCV, commonly known as the filter, can be used to apply it quickly and is detailed in[13] .

Instrumented gloves can be used, as described in , to extract the necessary image that will be used for training. When compared to applying filters to data obtained via video extraction, this helps save computation time for preprocessing and can provide us with data that is clearer and more precise.

We attempted to manually separate an image using color segmentation[14] algorithms, but the research paper points out that skin color and tone are highly reliant on lighting conditions because of which results we obtained from the segmentation we attempted were not great. Additionally, there are a vast array of symbols that need to be learned many of our projects, like the gesture, have a similar appearance to one another for the letter "V" and the number "2," therefore we determined that in order to separating the hand from a random background, we maintain the hand's background as a constant single color, eliminating the need to separate it based on skin tone. This would enable us to get better outcomes.

Gesture Classification:

Hidden Markov Models (HMM)[15] are employed to classify the gestures.

The dynamic elements of gestures are covered by this model.

By following the skin-color blobs that correspond to the hand into a body-facial space that is focused on the user's face, gestures can be recovered from a series of video shots. Recognizing the two categories of gestures—deictic and symbolic—is the aim. An indexing table with quick look-ups is used to filter the image. Skin color pixels are

collected into blobs after filtering. Blobs are statistical objects used to identify homogeneous areas. They are based on the position (x,y) and colorimetry (Y,U,V) of the skin color pixels.

In The Naive Bayes Classifier[16], a quick and efficient technique for recognizing static hand gestures, is employed. It is based on categorizing various gestures using geometric invariants that are obtained from segmented visual data. As a result, unlike many other recognition techniques, this technique is independent of skin tone. With a static background, the movements are taken from each frame of the video.

Segmenting, labeling, and extracting geometric invariants from the objects of interest is the initial stage. The next stage is to classify gestures using the K closest neighbor algorithm assisted with distance weighting algorithm (KNNDW)[17], which will then offer relevant information for a locally weighted Naive Bayes classifier[16].

The Institute of Automation Technology National Taipei University of Technology graduates Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen claim in their paper titled "Human Hand Gesture Recognition Using a Convolution Neural Network[18]" that they construct a skin model to extract the hand from an image before applying binary threshold to the entire image. Following the acquisition of the threshold picture, they calibrate it about the principal axis to center the image.

They calibrate the threshold picture about the major axis after acquiring it in order to center the image around it. To train and anticipate outputs, they feed this image into a convolutional neural network model. They trained their model using 7 hand gestures, and it now accurately reproduces those 7 movements with a 95% accuracy .

Feature Extraction and Representation :

A representation of an image as a 3D matrix with dimensions such as image height and width, and the value of each pixel as depth (1 for grayscale, 3 for RGB). These pixel values are then used to extract useful features using CNN.

Artificial Neural Networks :

An artificial neural network is a connection of neurons that mimics the structure of the human brain. Each connection of a neuron sends information to another neuron. Inputs are fed to the first layer of neurons where they are processed and sent to another layer of neurons called the hidden layer. After processing the information through multiple hidden layers, the information is passed to the final output layer.

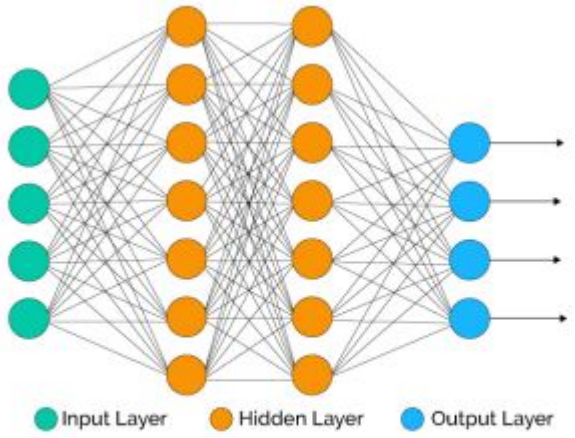


Figure 2.1 : Artificial neural networks

They are capable of learning and need training. There are different learning strategies:

1. Unsupervised learning
2. Supervised learning
3. Reinforcement learning

Convolution Neural Network :

Unlike regular neural networks, neurons in a CNN's layers are arranged in three dimensions: width, height, and depth. Neurons in a layer are not fully connected

to all neurons, but only to a small area (window size) of the layer before it. Moreover, the final output layer has dimensionality (number of classes) because at the end of the CNN architecture we reduce the big picture to a single vector of class scores.

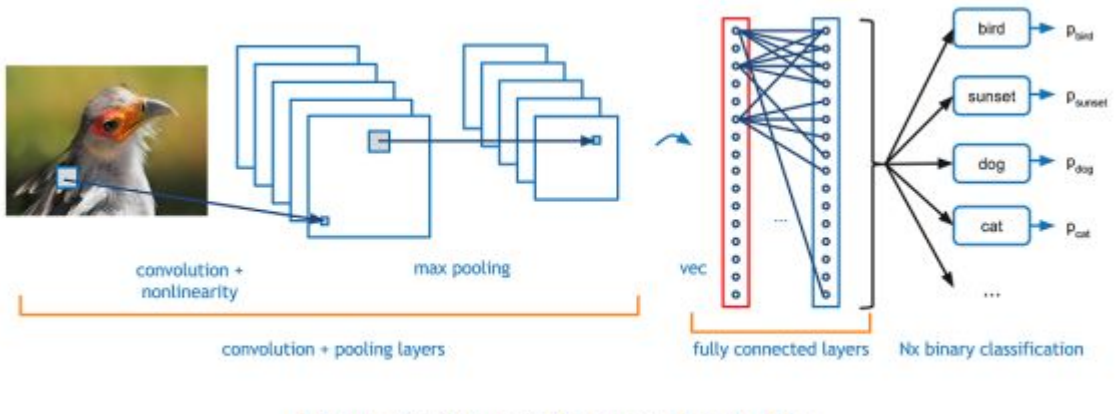


Figure 2.2: Convolution neural networks

1. **Convolutional Layers:** Convolutional layers take a small window size [typically length 5*5] that extends to the depth of the input matrix. A layer consists of a window size learnable filter. During each iteration, we moved the window by an increment [usually 1] and computed the dot product of the filter entry at the given position with the input value. We continue this process to create a two-dimensional activation matrix that reflects the response of this matrix at each spatial location. That is, the network learns filters and activates them when it sees visual features such as: B. Certainly oriented edges or spots of certain colors reduce the learnable parameters. There are two types of pooling:

a) Max pooling: Max pooling takes a window size [eg a window of size 2*2] and only takes up to 4 values. Close this window and

continue this process, and you will end up with an activation matrix that is half the original size.

b) Average pooling: Average pooling takes the average of all values in a window.

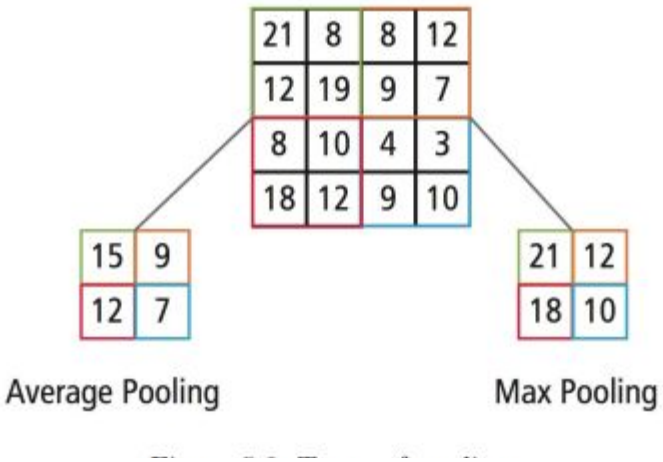


Figure 2.3: Types of pooling

3. **Fully Connected Layers:** In convolutional layers, neurons are only connected to local regions, whereas in fully connected regions all inputs are properly connected to neurons.

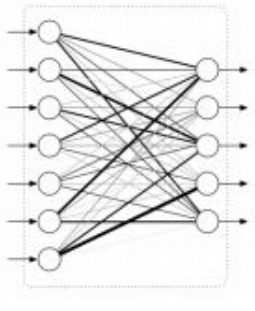


Figure 2.4 : Fully connected layer

4. **Final output layer:** After getting the values from the fully connected layer, connect them nicely to the last layer of [number equal to the total number of classes] neurons. This will predict the probability that each image is in a different class. .

TensorFlow :

TensorFlow is an open source software library for numerical computation. First we define the nodes of the computation graph, then the actual computation takes place within the session. TensorFlow is widely used in machine learning.

Keras:

Keras is a high-level neural network library written in Python that acts as a wrapper for TensorFlow. It is used to quickly build and test neural networks with minimal lines of code. It contains implementations of commonly used neural network elements such as layers, targets, activation functions, optimizers, and tools that facilitate manipulation of image and text data.

OpenCV :

OpenCV (Open Source Computer Vision) is an open source library of programming functions used for real-time computer vision. It is mainly used for image processing, video recording, and analyzing features such as face and object recognition. It is written in C++ with a primary interface, but bindings are available for Python, Java and MATLAB/O

CHAPTER 3

DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

- **Webcam based Real time Sign detection:** In this project for the ease of it's users. It can detect Sign detection in real time using the webcam. This function is quite user friendly as it doesn't have much hardware requirements.

- **Sign language to Text conversion :** This project converts the gesture to actual readable and meaningful text, which decreases the language barrier.

- **Realtime word suggestions:** Along with recognising and predicting the words in real time, it also suggest the possible words that can be formed using those letters, which helps the user to select the word he want to say more quickly.

3.2 Design Constraints

We experienced a lot of difficulties while working on the project. The dataset was the very first problem we ran into. We wanted to work with raw photos and square images alone, just like CNN in keras because it was much more convenient. We opted to create our own dataset because we couldn't discover any existing ones for it. The second challenge was choosing a filter that we could use to apply to our photographs, allowing us to extract the necessary properties of the images and use them as input for the CNN model. We experimented with a number of filters, such as binary threshold, canny edge detection, gaussian blur filter. Additional problems with the model's accuracy were encountered.

3.3 Analysis and Feature finalization subject to constraints

The features finalized for this project are very limited and minimal so that a clear set of features can be used and shown to the user which do not complicates the process and give the desirable outcome as well. The features are Webcam based real time sign detection, Sign language to text conversion and Real time word suggestions. Some more features can be added later on which includes converting the test output in speech which increase the factor of user friendliness significantly.

3.4 Design Flow

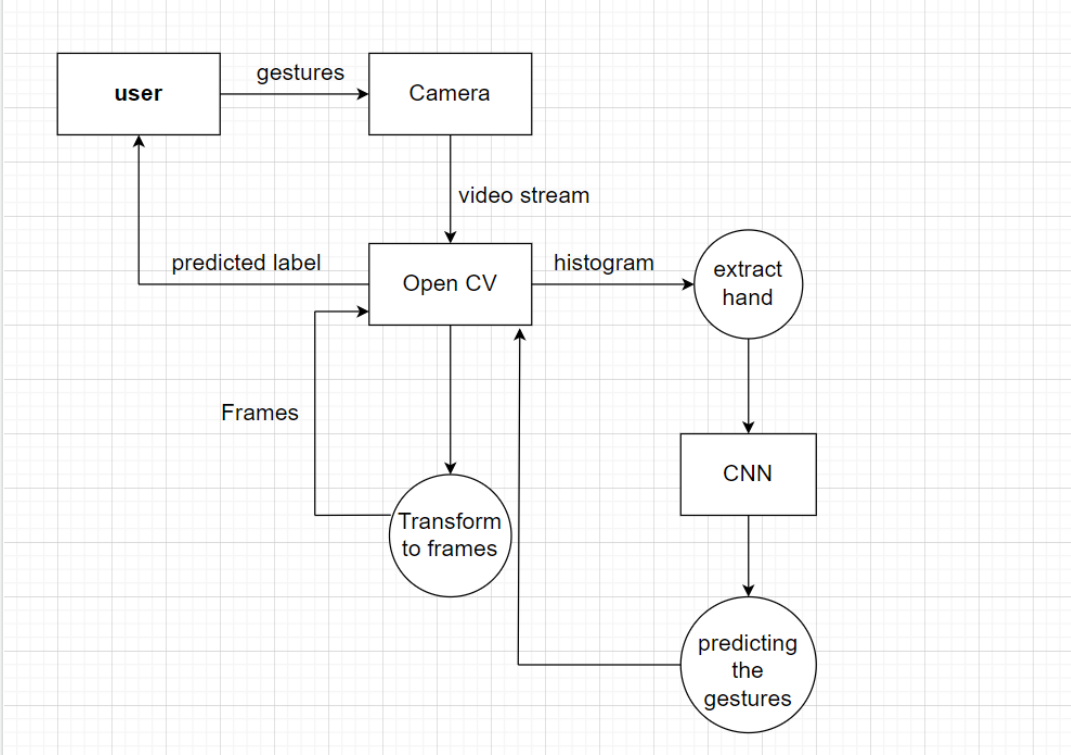


Figure 3.1 :DataFlow Diagram for Sign language recognition

3.5 Design selection

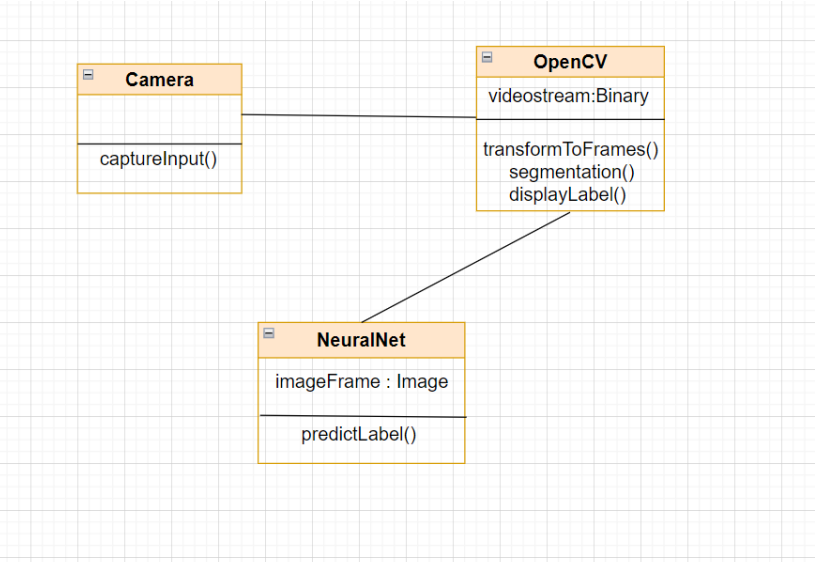


Figure 3.2 : Class diagram of Sign language recognition system

Figure 3.3: Use case diagram for sign language recognition

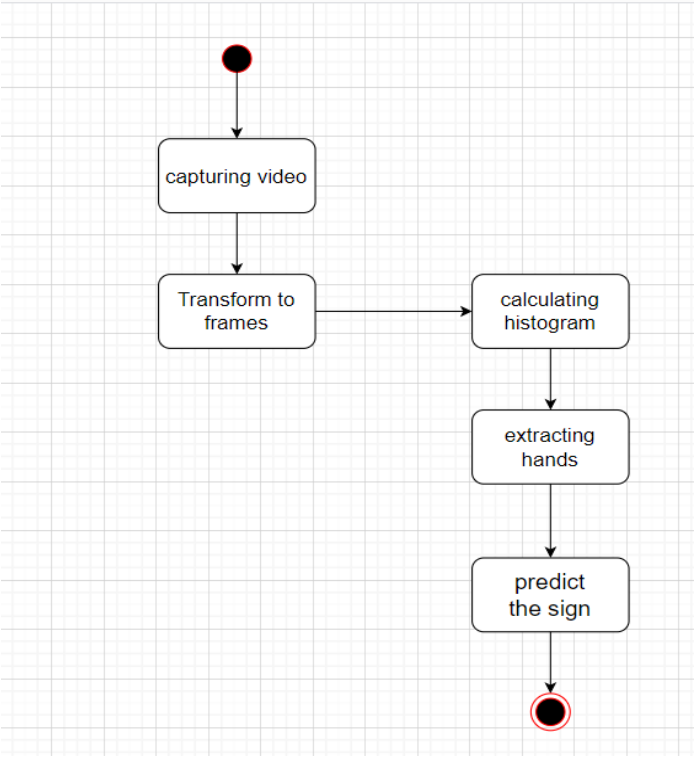


Figure 3.4: State chart diagram for Sign language recognition

3.6 Implementation plan/methodology

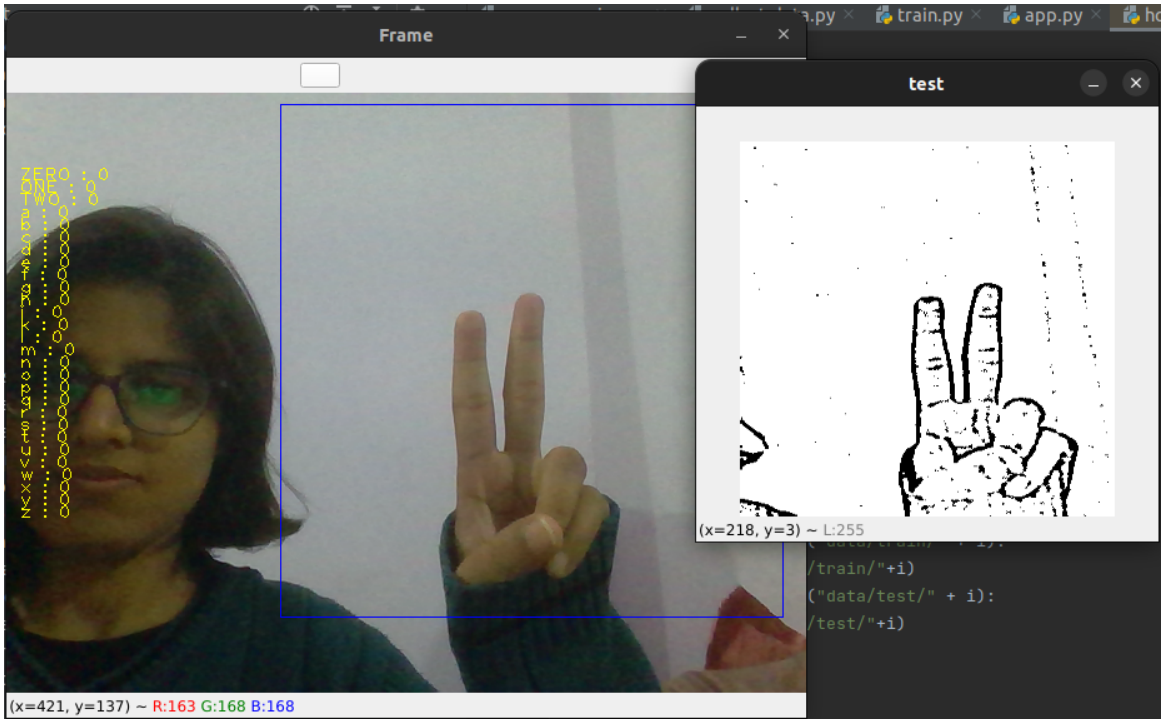
This project is based on a vision based approach. All the signs that are used in this project as inputs are formed with bare hands and with a blank screen as the background.

Data set Generation

For this project, I tried to find already created datasets, but could not find datasets in the form of raw images that met my requirements. All I found was a data set in the form of RGB values. So I decided to create my own dataset. Here are the steps to create the dataset:

I created a dataset using the Open Computer Vision (OpenCV) library. First, we acquired about 800 images of each symbol of ASL for training purposes, and about 200 images per symbol for testing purposes.First, capture each frame displayed by the machine's webcam. In each frame, define a region of interest (ROI) indicated by the square bounded by the blue as shown in the image below.

From this whole image, extract the ROI which is RGB and convert it to grayscale image as shown below. Finally, we apply a Gaussian Blur filter to the image. This allows you to extract various features of the image. After applying the Gaussian blur, the image looks like this:



Gesture Classification

This approach uses a two-layer algorithm to predict the final symbol for users.

Algorithm Level 1:

1. Apply gaussian blur filter and threshold to frame captured by opencv and get processed image after feature extraction.
2. This processed image is passed to a CNN model for prediction and if characters are detected over 50 frames, the characters are printed and are considered to form a word.
3. Spaces between words are considered spaces.

Algorithm Layer 2:

1. Once a is recognized, some set of symbols with similar results are recognized.
2. Then classify between these sets using a classifier created just for these sets

Layer 1:
CNN Model :

1. 1st Convolutional Layer:
Input image resolution is 128 x 128 pixels. It is first processed in the first convolution layer with 32 filter weights (3 x 3 pixels each). This produces 126 x 126 pixel images, one for each filter weight.

2. 1st pooling layer:
The image is downsampled using 2x2 max pooling. H. Keep the maximum value in a 2x2 square of the array. Therefore the image is scaled down to 63 x 63 pixels.

3. Second Convolutional Layer:
This 63 x 63 from the output of the first pooling layer is provided as input to the second convolutional layer. They are processed in a second convolution layer with 32 filter weights (3 x 3 pixels each). This will produce a 60 x 60 pixel image.

4. Second pooling layer:
The resulting image is downsampled again using the max pool of 2x2, reducing the down to a 30x30 image resolution.

5. 1st Tightly Connected Layer:
These images are taken as input to a fully connected layer containing 128 neurons and the output of the 2nd convolutional layer is transformed into an array of 30 x 30 x 32 = 28800 values will be The input for this level is an array of 28800 values. The output of this layer feeds the second adhesion layer. Use a 0.5 dropout layer to avoid overfitting.

6. Second Tightly Connected Layer:

Here the output of the first tightly connected layer is used as the input of a fully connected layer with 96 neurons.

7. Last layer:
The output of the 2nd tightly coupled layer serves as input for the last layer with the number of neurons as the number of classes to be classified (alphabet + space).

Activation function:

We used ReLu (Rectified Linear Unit) in each of the layers (convolutional and fully connected neurons). ReLu computes max(x,0) for each input pixel. This adds non-linearity to the formula and is useful for learning more complex functions. It helps speed up training by removing the vanishing gradient problem and reducing computation time.

Pooling Layer :

Apply Max Pooling to input image with pool size (2, 2) with relu activation function. This reduces the number of parameters, thus reducing computational cost and overfitting.

Dropout layer:

Overfitting problem. After training, the network weights are adjusted too much for the given training samples, and the network does not perform well on new samples. This layer is randomly "dropped" from this layer's group of activations by setting it to zero. A network should be able to provide correct classification or output for a given sample even if some activations fail .

Optimizer:

The Adam optimizer was used to update the model in response to the output of the loss function. Adam combines the advantages of his two extensions of two stochastic gradient

descent algorithms namely the Adaptive Gradient Algorithm (ADA GRAD) and Root Mean Square Propagation (RMSProp).

Layer 2:

A two-level algorithm is used to test and predict which symbols are similar to each other, and if you can recognize the symbol you see, you can get even closer. During my testing, I found that the following symbols are not displayed correctly and other symbols are printed as well.

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

So to handle the above case, we created three different classifiers to classify these sets:

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

Fingerspelling Sentence Formation Implementation:

1. Whenever the number of recognized characters exceeds a certain value and other characters are close to the threshold, output that character and append it to the current string (In this code, the value should be left at 50 and the difference threshold should be left at 20).
2. Otherwise, clear the current dictionary containing the recognition times for the current symbol to avoid the possibility of predicting the wrong character.
3. If the number of detected spaces (solid background) exceeds unique values and the current buffer is empty, no spaces are detected.
4. Otherwise, the end of the word is predicted by printing a space, and the current is added to the next sentence.

AutoCorrect features:

It uses the Python library Hunspell_suggest to suggest the correct alternatives for each (wrong) input word and displays the sequence of words corresponding to the current word. User can select additional words to replace the current word. records. This reduces misspellings and helps predict complex words

Training and Testing :

After converting the input image (RGB) to grayscale, applying Gaussian Blur to remove unwanted noise, and applying all the above operations, we train and test the model. The

prediction layer estimates the likelihood that an image falls into one of the classes. So the output is normalized between 0 and 1, and the sum of all values in each bin, equals 1. This was achieved with the softmax function.

First, the output of the prediction layer is a little far from the real value. For better results, we trained the network using tagged data. Cross entropy is a performance measure used in the classification. It is a continuous function that is positive for values not equal to the labeled value and zero only if equal to the labeled value. Therefore, we optimized by minimizing the cross-entropy to near zero. To do this, adjust the weights of the neural network in the network layer. TensorFlow has a built-in function to compute cross-entropy. Having found the cross entropy function, I optimized it with Gradient Descent using the best gradient descent optimizer called Adam Optimizer.

CHAPTER 4

RESULT ANALYSIS AND VALIDATION

4.1 Use of Modern tools in design and analysis

1. **Python**

   Python is a dynamically typed, garbage-collected programming language developed by Guido van Rossum in the late 1980s as an alternative to ABC. Python is designed to be easily readable by programmers. With many supporters and many libraries, you can implement Python to do everything from websites to scientific research. A computer program is developed using python language which is used to train the model based on the CNN algorithm .

2. **Pycharm**

   PyCharm is a purpose-built Python integrated development environment (IDE) that provides Python developers with a wide range of essential tools that are tightly integrated to create a working environment for productive Python, web, and data science development. PyCharm is cross-platform with Windows, macOS and Linux versions. When it comes to working on machine learning projects in Python, PyCharm is one of the most popular choices.

3. **Numpy**

   NumPy is an open source library for the Python programming language that adds support for large multidimensional arrays and matrices, as well as a large collection of high-level mathematical functions for manipulating these arrays.

4. **OpenCV**

   OpenCV (Open Source Computer Vision Library) is written in C/C++ for real-time computer vision and machine learning . Use multi-core processing and

hardware acceleration. OpenCV applications include ego-motion estimation, gesture recognition, facial recognition systems, and artificial neural networks.

5. **Keras**

   Keras is an open-source, cross-platform, and easy-to-use neural network library written in Python. Runs on TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML.

6. **Tensorflow** (as keras uses tensorflow in backend and for image preprocessing)

   TensorFlow is an open source library created by Google. Used to design, build, and train deep learning models. It is used to train the sign image from start to finish.

4.2 Discussion and report/results analysis:

4.2.1 Collection of data

There is no doubt that data collection is an integral part of this study, as our results depend heavily on it.We created a collect-data.py module for data collection.So, we have created our own data set and about 800 images of each symbol of ASL for training purposes and about 200 images per symbol for testing purposes.

```
preprocessing.py | collect-data.py | train.py | app.py | image_processing.py

import cv2
import numpy as np
import os
import string
# Create the directory structure
if not os.path.exists("data"):
    os.makedirs("data")
if not os.path.exists("data/train"):
    os.makedirs("data/train")
if not os.path.exists("data/test"):
    os.makedirs("data/test")
for i in range(3):
    if not os.path.exists("data/train/" + str(i)):
        os.makedirs("data/train/"+str(i))
    if not os.path.exists("data/test/" + str(i)):
        os.makedirs("data/test/"+str(i))

for i in string.ascii_uppercase:
    if not os.path.exists("data/train/" + i):
        os.makedirs("data/train/"+i)
    if not os.path.exists("data/test/" + i):
        os.makedirs("data/test/"+i)



# Train or test
mode = 'train'
directory = 'data/'+mode+'/'
if not os.path.exists("data/tes...
```

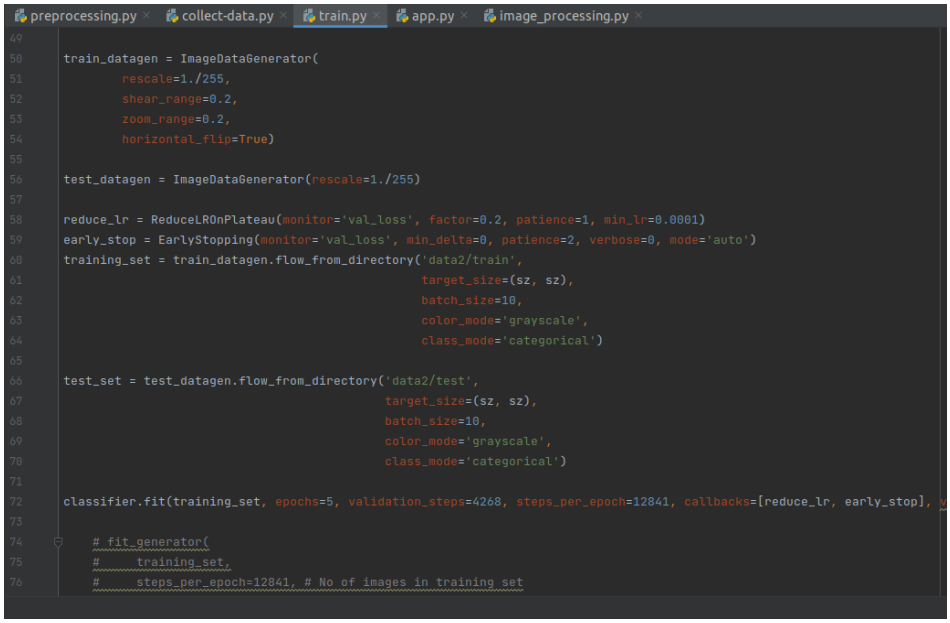
4.2.2 Pre-processing the data

In this step we preprocessed the image to be used later. Preprocessed image goes through RGB to Gray color conversion, then a layer of Gaussian blur and some threshold at the end. It makes only the boundary lines of hand visible and removing other unnecessary parts.

```
preprocessing.py | collect-data.py | train.py | app.py | image_processing.py

for (dirpath, dirnames, filenames) in os.walk(path):
    for dirname in dirnames:
        print(dirname)
        for(direcpath, direcnames, files) in os.walk(path+"/"+dirname):
            if not os.path.exists(path1+"/train2/"+dirname):
                os.makedirs(path1+"/train2/"+dirname)
            if not os.path.exists(path1+"/test/"+dirname):
                os.makedirs(path1+"/test/"+dirname)
            # num=0.75*len(files)
            num = 100000000000000000
            i=0
            for file in files:
                var += 1
                actual_path=path+"/"+dirname+"/"+file
                actual_path1=path1+"/"+"train2/"+dirname+"/"+file
                actual_path2=path1+"/"+"test/"+dirname+"/"+file
                img = cv2.imread(actual_path, 0)
                bw_image = func(actual_path)
                if i < num:
                    c1 += 1
                    cv2.imwrite(actual_path1, bw_image)
                else:
                    c2 += 1
                    cv2.imwrite(actual_path2, bw_image)

                i=i+1

        label=label+1
```

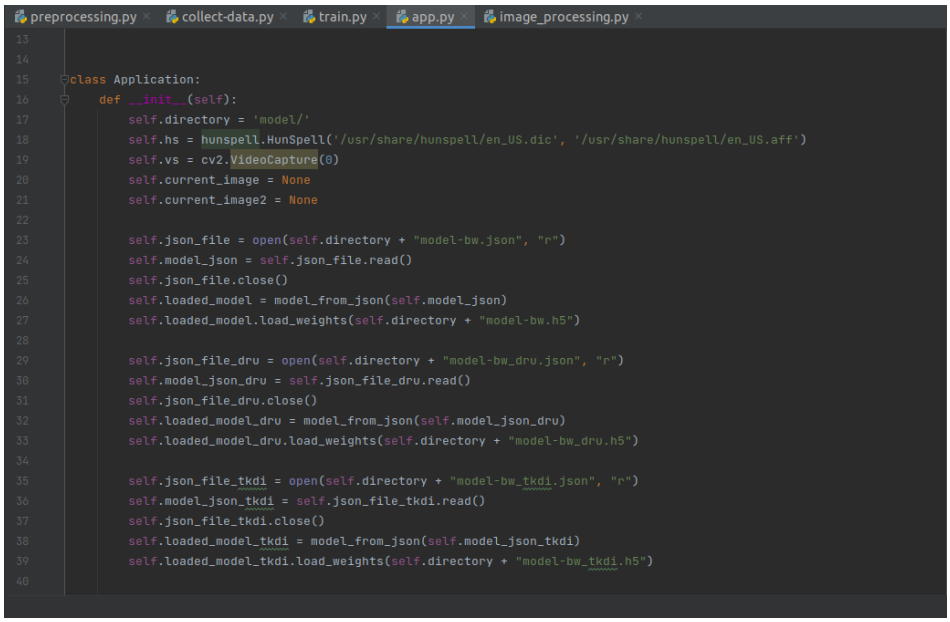
4.2.3 Training the model with data

In this step we trained the model using CNN and made it recognise the letter by the gestures.

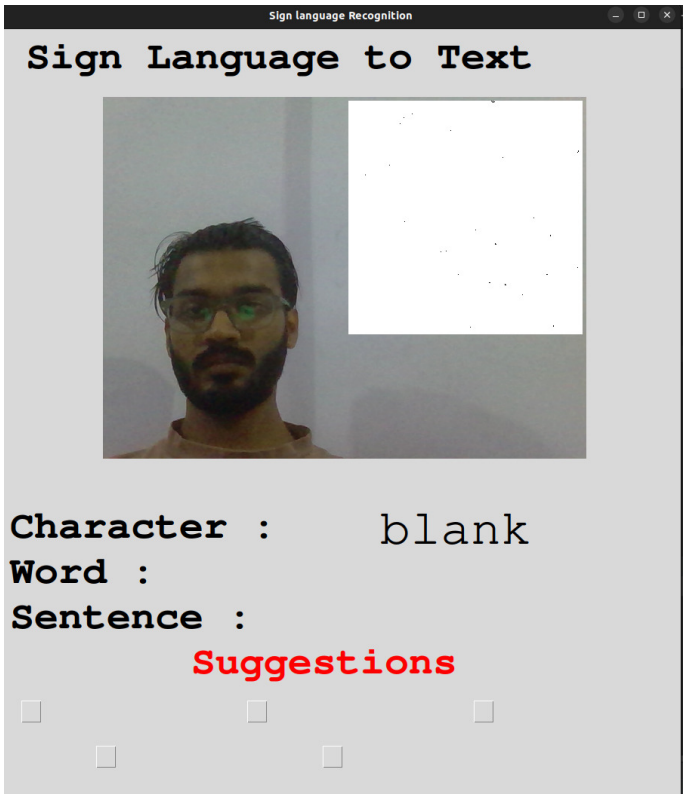


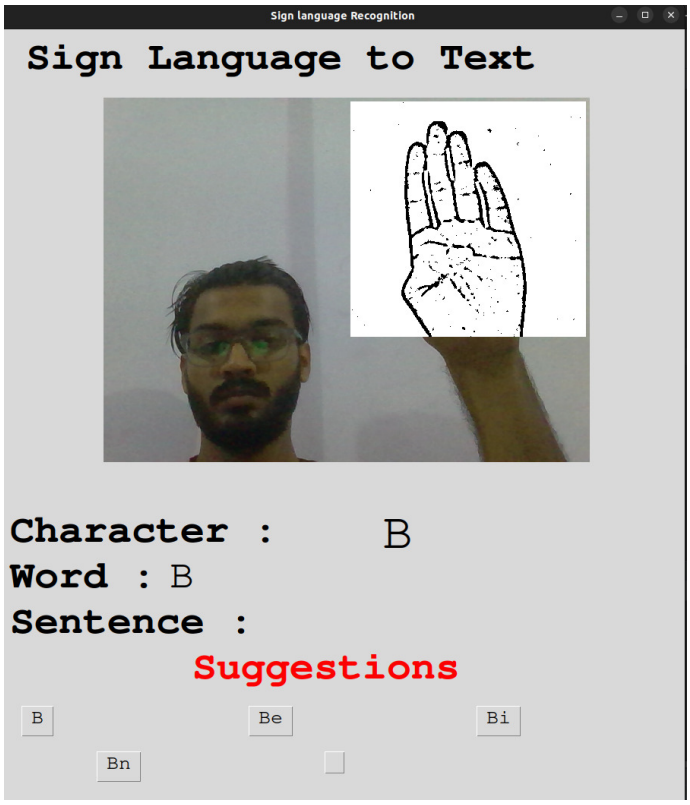
4.2.4 app.py

In this step we build the GUI where the input and output can be conceived from the same window which makes it easier for the user to use.



4.3 Output:


Sign language Recognition
Sign Language to Text
Character : blank
Word :
Sentence :
Suggestions


Sign language Recognition
Sign Language to Text
Character : B
Word : B
Sentence :
Suggestions
B
Be
Bi
Bn

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion :

We achieved 95.8% accuracy in our model using only layer 1 of the algorithm, and 98.0% accuracy using a combination of layers 1 and 2. Most research focuses on using devices such as his Kinect for hand recognition. [19] builds a Flemish Sign Language recognition system using a convolutional neural network and Kinect, achieving an error rate of 2.5%. In [20], recognition models were built using a hidden Markov model classifier and a vocabulary of 30 words, achieving an error rate of 10.90%. [21] achieves average accuracy of 86% for 41 static gestures in Japanese Sign Language. Using the Depth Sensors Map [22] achieved 99.99% accuracy with observed signers and 83.58% and 85.49% accuracy with new signers. They also used CNN for their detection system.

Note that our model does not use the background subtraction algorithm, but some of the models above do. So as soon as you try to implement background subtraction in your project, the precision may change. On the other hand, while most of the projects above use Kinect devices, our main goal was to create projects that could be used with readily available resources. Sensors like the Kinect are not only not readily available, they are also expensive to purchase for most viewers. The model uses a regular laptop webcam, so this is a big advantage. Below is the resulting confusion matrix.

.

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

Algo 1

		Predicted Values																								
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	A	147	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	2	0	0
	B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
	C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	0	0	0	3	10	0	0	0
Correct	G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
	I	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
Values	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
	S	0	0	0	0	1	0	0	0	0	0	0	0	0	0	10	0	0	0	133	0	0	0	0	8	0
	T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
	U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
	Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Algo 1 + Algo 2

In this report, a functional real-time sign language real-time recognition alphabet was developed. We achieved a final accuracy of 98.0% on our dataset. After validating and implementing his two layers of algorithms to predict symbols that are similar to each other, prediction can be improved. In this way, almost all symbols can be recognized, provided they are displayed correctly, the background is free of noise, and the lighting is sufficient.

5.2 Future Scope :

By experimenting with different background subtraction algorithms, we plan to achieve higher accuracy even with the complex background. We also hope to improve our preprocessing to predict gestures with better accuracy in low light. We are also planning to embed this application in a website which will make it more easy and accessible to use.

REFERENCES

[1] R. Rastgoo, K. Kiani, and S. Escalera, “Sign Language Recognition: A Deep Survey,” Expert Systems with Applications, vol. 164. 2021. doi: 10.1016/j.eswa.2020.113794.

[2] J. Prakash and U. K. Gautam, “Hand gesture recognition,” *International Journal of Recent Technology and Engineering*, vol. 7, no. 6, 2019, doi: 10.22214/ijraset.2022.43491.

[3] A. Mujahid *et al.*, “Real-time hand gesture recognition based on deep learning YOLOv3 model,” *Applied Sciences (Switzerland)*, vol. 11, no. 9, 2021, doi: 10.3390/app11094164.

[4] T. Ozseven, *Human-computer interaction*. 2019. doi: 10.1145/1226736.1226761.

[5] M. Oudah, A. Al-Naji, and J. Chahl, “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques,” *Journal of Imaging*, vol. 6, no. 8. 2020. doi: 10.3390/JIMAGING6080073.

[6] V. L. Averbukh, “Evolution of human computer interaction,” *Scientific Visualization*, vol. 12, no. 5, 2021, doi: 10.26583/SV.12.5.11.

[7] Y. Zhang, “Computer-assisted human-computer interaction in visual communication,” *Comput Aided Des Appl*, vol. 18, no. S1, 2021, doi: 10.14733/CADAPS.2021.S1.109-119.

[8] Y. Liao, P. Xiong, W. Min, W. Min, and J. Lu, “Dynamic Sign Language Recognition Based on Video Sequence with BLSTM-3D Residual Networks,” *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2904749.

[9] “Indian Sign Language Recognition Using Canny Edge Detection,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 3, 2021, doi: 10.30534/ijatcse/2021/131032021.

[10] J. Xu and X. Zhang, “A real-time hand detection system during hand over face occlusion,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 8, 2015, doi: 10.14257/ijmue.2015.10.8.29.

[11] D. G. Hong and D. Lee, “Vision-Based Hand Detection in Various Environments,” in *Lecture Notes in Mechanical Engineering*, 2020. doi: 10.1007/978-981-13-8323-6_29.

[12] F. Su, G. Fang, and J. J. Zou, “A novel colour model for colour detection,” *J Mod Opt*, vol. 64, no. 8, 2017, doi: 10.1080/09500340.2016.1261194.

[13] S. van der Walt *et al.*, “Scikit-image: Image processing in python,” *PeerJ*, vol. 2014, no. 1, 2014, doi: 10.7717/peerj.453.

[14] A. Marcireau, S. H. Ieng, C. Simon-Chane, and R. B. Benosman, “Event-based color segmentation with a high dynamic range sensor,” *Front Neurosci*, vol. 12, no. APR, 2018, doi: 10.3389/fnins.2018.00135.

[15] Z. Ghahramani, “An introduction to hidden Markov models and Bayesian networks,” *Intern J Pattern Recognit Artif Intell*, vol. 15, no. 1, 2001, doi: 10.1142/S0218001401000836.

[16] S. Taheri and M. Mammadov, “Learning the naive bayes classifier with optimization models,” *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 4, 2013, doi: 10.2478/amcs-2013-0059.

[17] M. Sharma and M. Biswas, “KLT-CRKCN: Hyperspectral Image Classification via Karhunen Loeve Transformation and Collaborative Representation-Based K Closest Neighbor,” *Wirel Pers Commun*, vol. 123, no. 4, 2022, doi: 10.1007/s11277-021-09292-4.

[18] H. I. Lin, M. H. Hsu, and W. K. Chen, “Human hand gesture recognition using a convolution neural network,” in *IEEE International Conference on Automation Science and Engineering*, 2014, vol. 2014-January. doi: 10.1109/CoASE.2014.6899454.

[19] 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

[20]Zaki, M.M., Shaheen, S.I.: Sign language recognition using a combination of new vision based features. Pattern Recognition Letters 32(4), 572–577 (2011)

[21] 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

[22]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)