A Mini Project Report
on

GESTURE AI

Submitted in partial fulfillment of the

Requirements for the award of degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Pippari Shashank Mourya
(19H61A0537)

Panugothu Laxmi Archana
(19H61A0534)

Jaganti Hemanth Kumar
(19H61A0525)

Under the Guidance of

Dr. P SRILATHA, M. Tech., Ph.D.

(Asst. Professor, Department of CSE)

Department of Computer Science and Engineering
ANURAG GROUP OF INSTITUTIONS
(Formerly CVSR College of Engineering)
(An Autonomous Institution, Approved by AICTE and NBA Accredited)
Venkatapur (V), Ghatkesar (M), Medchal(D)., T.S-500088
(2019-2023)

Anurag Group of Institutions

An Autonomous Institution

(Formerly CVSR College of Engineering)

Venkatapur (V), Ghatkesar (M), Ranga Reddy (Dist.)

Ph: 08499953666, 08499963666. www.anurag.edu.in

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **" GESTURE AI "** being submitted by **Pippari Shashank Mourya, Panugothu Laxmi Archana, Jaganti Hemanth Kumar** bearing the Hall Ticket number **19H61A0537, 19H61A0534, 19H61A0525** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to **Anurag Group of Institutions (Formerly CVSR College of Engineering)** is a record of bonafide work carried out by them under my guidance and supervision from June 2022 to October 2022.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Internal Guide **External Examiner**

Dr. P Srilatha
Asst. Professor, Department of CSE

Dr. G. Vishnu Murthy,
Professor & Head, Dept. of CSE

ACKNOWLEDGEMENT

It is our privilege and pleasure to express profound sense of respect, gratitude and indebtedness to our guide **Dr. P Srilatha, Asst Professor**, Dept. of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his/her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere gratitude to **Dr. G. Vishnu Murthy, Professor & Head**, Department of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his suggestions, motivations, and co-operation for the successful completion of the work.

We extend our sincere thanks to **Dr. V. Vijaya Kumar**, **Dean,** Research and Development, Anurag Group of Institutions, for his encouragement and constant help.

Pippari Shashank Mourya
19H61A0537

Panugothu Laxmi Archana
19H61A0534

Jaganti Hemanth Kumar
19H61A0525

DECLARATION

We hereby declare that the project work entitled "**GESTURE AI**" submitted to the **Anurag Group of Institutions(Formerly CVSR College of Engineering)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Dr. P Srilatha, Assistant Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

Pippari Shashank Mourya
19H61A0537

Panugothu Laxmi Archana
19H61A0534

Jaganti Hemanth Kumar
19H61A0525

Date:

ABSTRACT

Hand gesture is one of the methods used in sign language for non-verbal communication that is used for deaf and dumb people. It is commonly used by deaf & dumb people who have hearing problems or speech problems to communicate among themselves or with normal people. Many sign language systems have been developed by many makers around the world but we found that they have drawbacks like they are neither flexible nor cost-effective for the end users. Hence in the proposed system introduced software which presents a system prototype that is able to automatically recognize sign language to help deaf and dumb people to communicate more effectively with each other or normal people. As in the field of research Pattern recognition and Gesture recognition are the developing fields. So, Being a significant part in nonverbal communication, hand gestures play a key role in our daily life. The proposed system provides us an innovative, natural, user-friendly way of communication with the computer which is more familiar to human beings. By considering the similarities of human hand shape with four fingers and one thumb, the proposed system aims to present a real time system for recognition of hand gesture on the basis of detection of some shape based features like orientation, Centre of mass centroid, fingers status, thumb in positions of raised or folded fingers of hand.

Keywords — sign language recognition, hand gesture recognition

CONTENTS

1. INTRODUCTION

Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty with their gestures, as only very few of those are recognized by most people. Hence there is a need of a system which recognizes the different signs, gestures, and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

1.1. MOTIVATION

The 2011 Indian census cites roughly 1.3 million people with “hearing impairment”. In contrast to that numbers from India’s National Association of the Deaf estimates that 18 million people roughly 1 per cent of Indian population are deaf. These statistics formed the motivation for our project. As these speech impairment and deaf people need a proper channel to communicate with normal people there is a need for a system.

Sign language is a form of communication used by people with impaired hearing and speech. People use sign language gestures as a means of non-verbal communication to express their thoughts and emotions. Sign Language is a subset of gestures or signs made with fingers, hands, arms, eyes, and head, face etc. Each gesture in sign language has a meaning assigned to it. Understanding sign language is nothing but understanding the meaning of these gestures. There exists a problem in communication when a person who completely relies on this gestural sign language for communication tries to converse with a person who does not understand the sign language. Hand gestures are used to detect the different signs, for etc.., their are hand gestures for traffic signals and some researchers are working on it, static image detection. Many methods and framework come into existence like artificial neural network, convolutional neural network, deep learning etc..,

This motivated us to build a deep learning based sign language recognition framework, CNN are ideal for image classification problems. Not all normal people can understand sign language of impaired people. Due to which there is a need of a system which recognizes the different signs, gestures, and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people. Sign language is the only tool of communication for the person who is not able to speak and hear anything.

Sign language is a boon for the physically challenged people to express their thoughts and emotion. In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabets and gestures in sign language. With the help of computer vision and neural networks we can detect the signs and give the respective text output. The idea is to make computers to understand human language and develop a user-friendly human computer interface. Hence our project is aimed at converting the sign language gestures into text that is readable for normal people.

1.2. PROBLEM DEFINITION

To design a real time software that will help to recognize the hand gesture using deep learning techniques and CNN based network. The project aims to determine human gestures by creating an HCI.

1.3. OBJECTIVES

The objective of this project is to develop an intelligent system which can act as a translator between the sign language and the spoken language dynamically and can make the communication between people with hearing impairment and normal people both effective and efficient.

1. Analyze gestures with different image extraction tools.
2. Develop a classification system using CNN classifier.

2. LITERATURE SURVEY

Literature review shows that there have been several approaches and methods to solve this problem some using various algorithms.

In the paper **“Sign Language Translation”** published in 2020 proposed a system for implementing computer vision which can take the sign from the users and convert them into text in real time. This system is designed and implemented to recognize sign language gestures. The four modules of this system are Image capturing, Preprocessing, Classification and Prediction. The primary focus of the system is to support detecting gestures in dynamic background condition. The pixel values of each image are stored in csv file, which reduces the memory requirement of the system, is used as the dataset. Also, the accuracy of prediction is high when csv dataset is used. To achieve this, the frames are preprocessed and converted to gray scale image and then background subtraction algorithm is used. After collecting and processing the image dataset, they must be classified. Convolutional neural network is used to analyze and classify visual imagery. CNN is regularized versions of multilayer perceptron. It consists of convolutional layer, pooling layer, flattening and fully connected layer along with activation functions, where Convolution layer is performed an image to identify certain features in an image. A convoluted image can be too large or too small and therefore needs to be reduced without losing its features, when max pooling layer max value is selected and in min pooling layer min value is selected from a particular region. This layer transforms multi-dimensional matrix to 1-d array so that it can be fed into the classifier. The proposed system translates the sign gestures to text using CNN. The accuracy of the model is 99.91%. Although the facial expressions express a lot during communication, the system does not focus on facial expressions. The accuracy of the model was less with poor lighting. As future enhancements, more dynamic video signs can be trained involving facial features and expressions.

In the paper **" American Sign Language Recognition using Deep Learning and Computer Vision "** published in 2018 the authors used two models a CNN and a RNN to create a vision-based application which offers sign language translation to text thus, aiding communication between signers and non-signers. They created a data set of hundred different signs from the American Sign Language data set. Each sign is performed five times by a single signer in varying lighting conditions and speed of signing. The videos were recorded on an iPhone 6 camera on 60fps and at 720p resolution. Each video was broken down by frame to images and trimmed to 300 frames and then augmented to increase the data set for each sign to 2400 images. The images were re sized and rotated at random as part of the augmentations. The data set was further divided into training and test data sets, with 1800 as part of the training and the remaining as the test data set. The CNN model extracted temporal features from the frames which was used further to predict gestures based on sequence of frames.

The CNN model used was Inception, which was a model developed by Google for image recognition as is widely regarded as the most image recognition neural network which exists right now. Then, by using a LSTM (Long Short-Term Memory), a RNN (Recurrent Neural Network) model, to extract temporal features from the video sequences via two methods: Using the outputs from the SoftMax and the Pool layer of the CNN, respectively. The camera is used to record the hand gestures and then the videos are preprocessed to frame sequences which is fed individually to the CNN for two possible outputs. The global pool layer gives us a 2048 sized vector, which possibly allows for more features to be analyzed by the RNN. The feature sequence is then fed to a Long-Short Term Memory (LSTM) allowing longer time dependencies. RNN's suffer from the vanishing/exploding gradient problem, LSTM's deal with the problem allowing for higher accuracies on longer sequences of data. The outputs of the SoftMax Layer and the Max Pooling layer and feed it to the RNN. The gesture segments identified and processed by the CNN are classified by the LSTM into one of the gesture classes using sequence data. They trained the LSTM on the outputs from the CNN SoftMax layer and the Pool layer and compared the results. The CNN gave the accuracy of 96% but the RNN gave 57%. The model also suffered from loss of accuracy with the inclusion of faces. The model also performed poorly when there was variation in clothing.

In the paper **"Indian Sign Language Translator Using Gesture Recognition Algorithm"** the authors developed an algorithm that will translate the ISL into English and implemented a system. This system translates gestures made in ISL into English. The gestures that have been translated include numbers, alphabets, and few phrases. The algorithm first performs data acquisition, then the pre-processing of gestures is performed to track hand movement using a combinational algorithm, and recognition is done using template matching. The database of the gestures is self-created. Hence system begins with data collection. Data is acquired from signers who are deaf and mute by birth. The signs recorded include the numbers, alphabets, and some phrases. The complete database includes total 130,000 videos out of which 58,000 videos are tested for checking the performance of the system. The developed system makes use of a combination of three methods for feature extraction.

In this system before proceeding for feature extraction, first the raw data of gestures is trained to give out only useful information. The frames are cropped then two subsequent frames are subtracted from each other for obtaining the difference image. In order to achieve segmentation of threshold image. Once the preprocessing of gesture was done, hand motion is traced. This step is important because gesture has movement of hand as an integral part of it. In order to track the hand movement a combination algorithm was used which takes into consideration hand motion, skin color and the boundary of hand of the signer. To extract the external boundary points of a hand shape the contour following algorithm “Image Compression Using Learned Vector Quantization" was used. To represent the boundary points, first the Fourier series were calculated using Fast Fourier Transform (FFT) algorithm. For feature extraction purpose, 28 Fourier descriptors per each frame of video gesture are considered. Before selecting 28 descriptors, the system was implemented using 30 descriptors as well as 25 descriptors. The best results were obtained using 28 x 28 descriptors. After implementing a feature extraction using FDs the extracted data that was obtained was too large.

In order to store the reference codebook, compression was required. This compression has been achieved using vector quantization. Vector quantization (VQ) is a non-uniform and many-to-one mapping, responsible for lossy compression. It was based on equivalence relation. The vector quantization scheme can be divided into three parts: the codebook generation process, the encoding procedure and the decoding procedure. The most important factor of vector quantization technique is to design a good codebook. The type of vector quantization that was employed was popularly known as Linde- Buzo- Gray (LBG) type of vector quantization. Once the Feature extraction is done, the next step is to match the extracted features of a testing sequence with the previously saved training reference codebook vectors. This system makes use of a simple Euclidean Distance method. The system was implemented for 10 users and the gestural data of 10 numbers, 26 alphabets and 10 different phrases. The results show overall accuracy of the system to be 92.91% when Numbers, Alphabets and Phrases are considered together. Where the accuracy of Alphabets individually was 85.73%, accuracy of numbers individually is 95.5% and accuracy of phrases is as high as 97.5%. The human facial features or gestures are not implemented in this model, which plays a major role in non-verbal communication.

In the paper **“Sign Language Recognition”** published in 2015 the authors aimed at building a machine learning model that will be able to classify the various hand gestures used for fingerspelling in sign language. Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Various machine learning algorithms are used, and their accuracies are recorded.

Two data sets used ASL dataset and ISL data set. ASL dataset created by B. Kang et al is used. It is a collection of 31,000 images, 1000 images for each of the 31 classes. No standard dataset for ISL was available.

For this project, various classification algorithms are used: SVM, k-NN and CNN. Feature extraction algorithms are used for dimensionality reduction When the input to the algorithm is too large to be processed and is suspected to be redundant then it can be converted into a reduced set of features. Feature extraction algorithms: PCA, LBP, and HOG, are used alongside classification algorithms for this purpose. This reduces the memory required and increases the efficiency of the model.In k-NN classification, an object is classified by a majority vote of its neighbors, this is assigned to a class then the output of the algorithm is a class membership. In SVM, each data point is plotted in an n-dimensional space this differentiates the classes to the best. Convolutional Neural Networks used to process data that have a grid-like topology, e.g., images that can be represented as a 2-D array of pixels. A CNN model consists of four main operations: Convolution, Non-Linearity, Pooling and Classification. Convolution is to extract features from the input image. It is an element-wise operation that replaces all negative pixel values in the feature map by zero. The concept of Transfer learning is used here, where the model is first pre-trained on a dataset that is different from the original. The image dataset was converted to a 2-D array of pixels. The array was flattened and normalized.

A confusion matrix gives the summary of prediction results on a classification problem. However, pre-training has to be performed with a larger dataset in order to show increase in accuracy. We were able to achieve maximum accuracy of 71.88% with SVM+HOG for ISL dataset using depth images dataset when 4 subjects were used for training and a different subject for testing.

3. ANALYSIS

3.1. EXISTING SYSTEM

Identification of sign gesture is mainly performed by the following methods:

1. **Glove-based method** in which the signer has to wear a hardware glove, while the hand movements are getting captured.
2. **Vision-based method**, further classified into static and dynamic recognition. Statics deals with the detection of static gestures(2d-images) while dynamic is a real-time live capture of the gestures. This involves the use of the camera for capturing movements.

3.2. PROPOSED SYSTEM

Our proposed system is sign language recognition system using convolution neural networks which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus our system is more robust in getting exact text labels of letters.

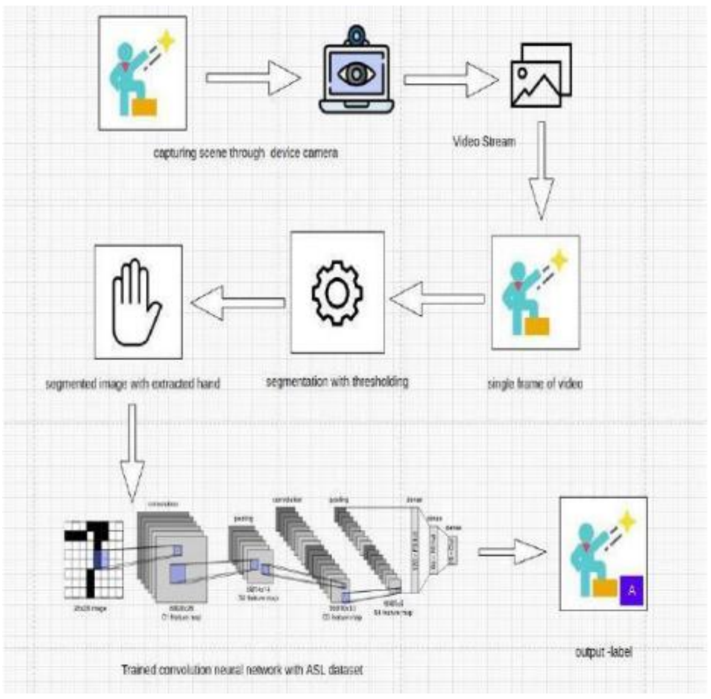

capturing scene through device camera
Video Stream
segmented image with extracted hand
segmentation with thresholding
single frame of video
output -label
Trained convolution neural network with ASL dataset

Fig 3.1. System Architecture

3.3. SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

Operating System: Windows, Mac, Linux
SDK: OpenCV, TensorFlow, Keras, Numpy
Camera: Good quality,3MP
Ram: Minimum 8GB or higher
Processor: Intel Pentium 4 or higher
HDD: 10GB or higher
Monitor: 15” or 17” color monitor
Packages: CVZone

3.3.1. PURPOSE

This project aims to determine the gesture from the captured image in real time. Based on the signs from ASL, representing the English alphabet, a dataset is created. We used the model we trained in Teachable Machine and integrated it into our code. The python module cvzone is used. This project's main goal is to let us know the gesture's symbol from the image captured in real-time irrespective of the angle of the hand. So, that the deaf and dumb can interact with people who don’t know sign language.

3.3.2. SCOPE

Image Processing part should be improved so that System would be able to communicate in both directions i.e., it should be capable of converting normal language to sign language and vice versa. We will try to recognize signs which include motion. Moreover we will focus on converting the sequence of gestures into text i.e. word and sentences and then converting it into the speech which can be heard. The scope of this project can be extended to many facets of the society. The enrichment of daily life of the people who use sign language as a means of expression by enabling a means of translation between them and the normal people. Consider having a integrated video conferencing platform which enables real-time translation of sign language. Or a small animated bot which translates textual language to sign language.

3.3.3. OVERALL DESCRIPTION

Sign language is the tool used to communicate with the impaired people, where they use hand signs and gestures to communicate with normal people. We had captured the images and created a dataset using computer vision and numpy to store the images. The images are first superimposed with a skeleton generated by the cv-zone module then the image is saved temporarily which is later cropped and stored into the database. We then used the Google Teachable Machine to generate a model using the dataset. The generated model was later integrated into the code. The generated model was a CNN model more specifically a MobileNet model. Later for the testing we captured real-time images which are fed into the testing code. The result was indicated at the top-left of a box generated around the hand.

4. DESIGN

4.1. UML DIAGRAMS

UML stands for Unified Modeling Language. Taking SRS document of analysis as input to the design phase drawn UML diagrams. The UML is only language so is just one part of the software development method. The UML is process independent, although optimally it should be used in a process that should be driven, architecture-centric, iterative, and incremental. The UML is language for visualizing, specifying, constructing, documenting the articles in a software-intensive system. It is based on diagrammatic representations of software components. A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of the system. A modeling language such as the UML is thus a standard language for software blueprints. The UML is a graphical language, which consists of all interesting systems. There are also different structures that can transcend what can be represented in a programming language. These are different diagrams in UML.

4.1.1. USE CASE

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behaviour of the system as seen from the actor’s point of view. It describes the function provided by the system as a set of events that yield a visible result for the actor. Purpose of Use Case Diagrams The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams 41 (activity, sequence, collaboration, and Statechart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purposes of use case diagrams can be said to be as follows − Used to gather the requirements of a system. Used to get an outside view of a system. Identify the external and internal factors influencing the system. Show the interaction among the requirements are actors. Functionalities to be represented as use case Actors Relationships among the use cases and actors. 42 Use case diagrams are drawn to capture the functional requirements of a system.

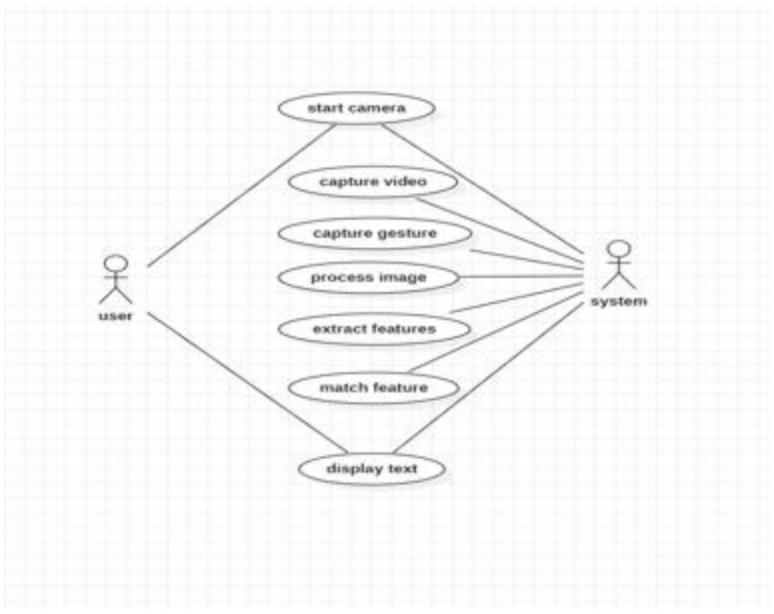


Fig 4.1. Use case diagram

4.1.2. CLASS DIAGRAM

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagram describe the different perspective when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

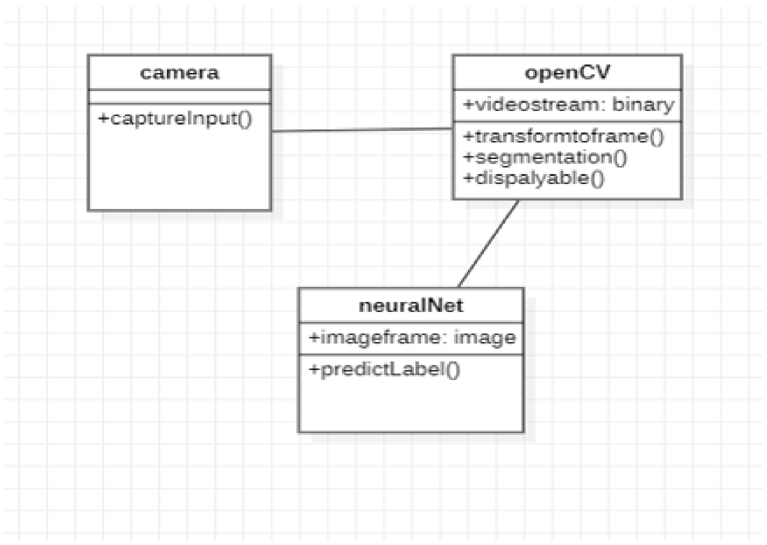


Fig 4.2. Class Diagram

4.1.3. SEQUENCE DIAGRAM

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension(time) and horizontal dimension (different objects). Objects: Object can be viewed as an entity at a particular point in time with specific value and as a holder of identity. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response.

Asynchronous calls are present in multithreaded applications, event-driven applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML). Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. If an object is destroyed (removed from memory), an X is drawn on bottom of the lifeline, and the dashed line 48 ceases to be drawn below it. It should be the result of a message, either from the object itself, or another. A message sent from outside the diagram can be represented by a message originating from a filled-in circle (found message in UML) or from a border of the sequence diagram (gate in UML). UML has introduced significant improvements to the capabilities of sequence diagrams. Most of these improvements are based on the idea of interaction fragments which represent smaller pieces of an enclosing interaction. Multiple interaction fragments are combined to create a variety of combined fragments, which are then used to model interactions that include parallelism, conditional branches, optional interactions.

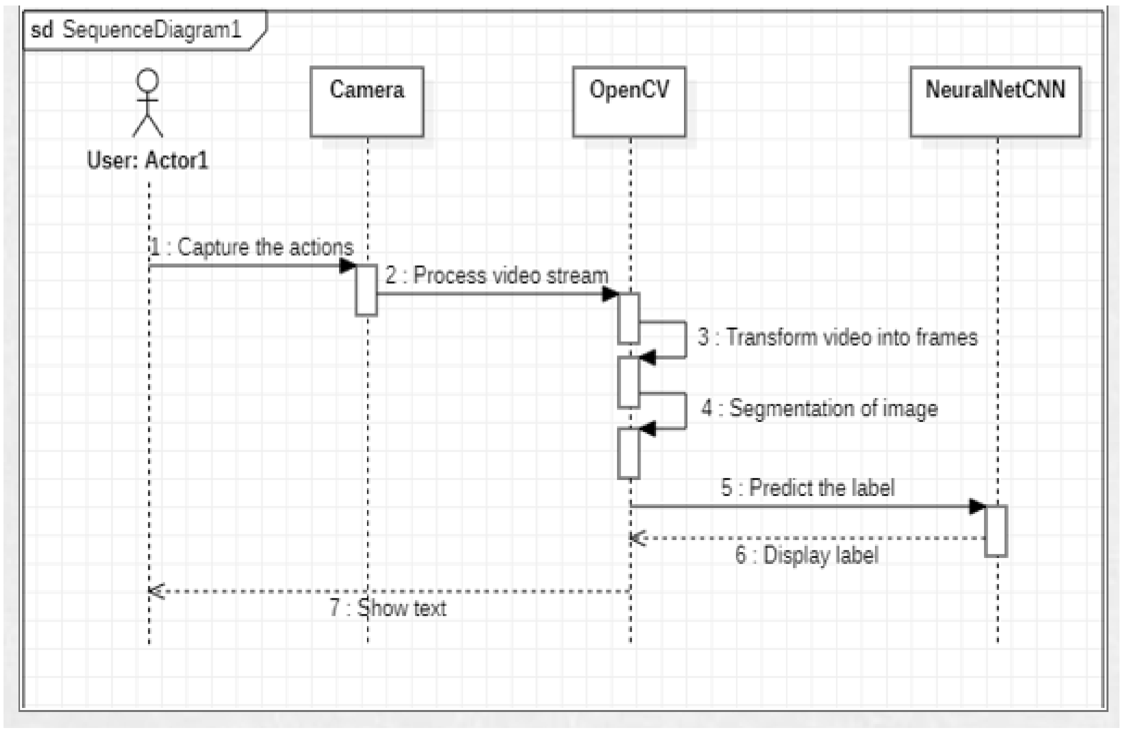


Fig 4.3. Sequence Diagram

4.1.4. STATE CHART DIAGRAM

A state chart diagram describes a state machine which shows the behaviour of classes. It shows the actual changes in state not processes or commands that create those changes and is the dynamic behaviour of objects over time by model1ing the life cycle of objects of each class. It describes how an object is changing from one state to another state. There are mainly two states in State Chart Diagram:1. Initial State 2. Final-State. Some of the components of State Chart Diagram are: State: It is a condition or situation in life cycle of an object during which it's satisfies same condition or performs some activity or waits for some event. Transition: It is a relationship between two states indicating that object in first state performs some actions and enters into the next state or event. Event: An event is specification of significant occurrence that has a location in time and space.

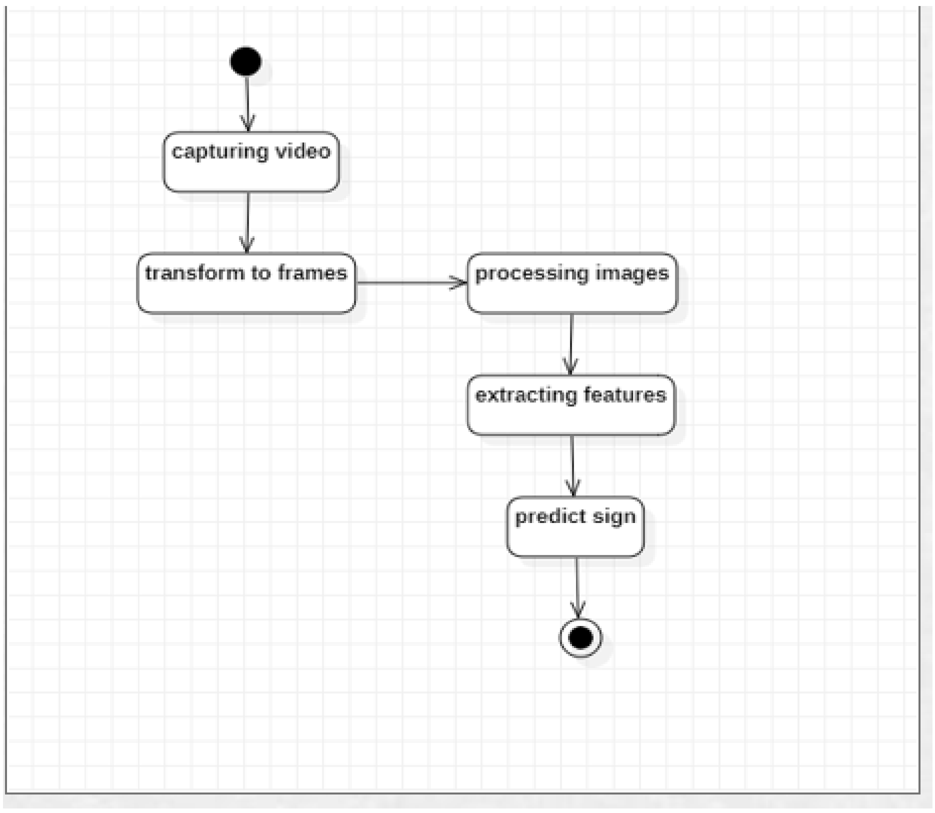


Fig 4.4. State Chart Diagram

5. IMPLEMENTATION

5.1. MODULES

We have divided our project into three modules.

1. Data collection module: Through this model we collected our own dataset.
2. Training module: This model is used to integrate the model we trained from google teachable machine into the code.
3. Testing module: The testcase is loaded into this model to know the result.

5.2. MODULE DESCRIPTION

5.2.1. DATA COLLECTION MODULE:

This module used to collect dataset form the real world. It used cvzone modules like HandTracking Module which has findhands() function,used to identify hands and extract them from live video stream.The extracted hand is fitted into a fixed size frame of 300*300 pixels using crop function and can be customised for different hand sizes and different hand signs like width and height. In order to save the images in respective folder ,on pressing control+s key the image on the screen can directly save to respective folder in described file path. All images are stored in different folders using labels as their class name and number of images each class can vary according to the movement of the hand and capturing positions. We will be having a live feed from the video cam and every frame that detects a hand in the ROI (region of interest) created will be saved in a directory (gesture directory) that contains folders containing images captured using the dataCollection.py.

5.2.2. TRAINING MODULE:

In this module, to train our model, we used Google Teachable Machine, these models use a technique called transfer learning. There’s a pretrained neural network, and when you create your own classes, you can sort of picture that your classes are becoming the last layer or step of the neural net. Specifically, both the image and pose models are learning off of pretrained MobileNet models.

So we uploaded all our captured dataset into Google Teachable Machine class-wise, and started to train the model. Based on epoch size and training time ,the model is trained. These models give us output as Keras module which can be used for testing implemented using python. The trained model hence downloaded contains .h5 file and labels.txt file.

5.2.3. TESTING MODULE:

In this moduel, the downloaded trained model is tested using python against real world images that will be captured using the same dataCollection.py module and after detecting the

Hand from it ,we just added a bounding frame to it which is used to display the result, on the top-left corner of the box. The classified result is displayed on the text area provided. All this happens only in live video streaming .The excepted output is displayed on to the screen on the hand. This is tested against various gestures through live video from where the hand is extracted and classified into correct class.

5.3. INTRODUCTION TO TECHNOLOGIES USED

5.3.1. CVZONE:

This is a computer vision package that makes it easy to run Image processing and AI functions. At the core it uses OpenCV and Media pipe libraries. One can simply use pip to install the latest version of cvzone. (pip install cvzone). It is used for Face Detection, Hand Tracking, Pose Estimation, Face Mesh Detection, Stack Images, Corner Rectangle and FPS.

5.3.2. ABOUT GOOGLE TEACHING MACHINE:

Google Teaching Machine uses a technique called transfer learning. There is a pretrained neural network, and when you create your own classes, you can sort of picture that your classes are becoming the last layer or step of the neural net. Specifically, both the image and pose models are learning off pretrained mobile-net models, and the sound model is built on Speech Command Recognizer.

5.4. SAMPLE CODE

5.4.1. DATA COLLECTION:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset = 20
imgSize = 300
folder = "Data/A"
counter = 0
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand[‘bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
       if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        counter += 1
        cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',imgWhite)
        print(counter)
```

5.4.2. TESTING CODE:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")
offset = 20
imgSize = 300
# folder = "Data/C"
# counter = 0
labels = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N", "O",
"P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y"]
while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]
        x, y, w, h = hand['bbox']
        imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
        cv2.rectangle(imgOutput, (x - offset, y - offset - 50),
                      (x - offset + 90, y - offset - 50 + 50), (255, 0, 255), cv2.FILLED)
        cv2.putText(imgOutput, labels[index], (x, y - 26),
cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
        cv2.rectangle(imgOutput, (x - offset, y - offset),
                      (x + w + offset, y + h + offset), (255, 0, 255), 4)
        cv2.imshow("ImageCrop", imgCrop)
        cv2.imshow("ImageWhite", imgWhite)
    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

6. TEST CASES

We used various hand geustures to know the result of the project. These hand gestures all fall under the below mentioned test cases with their results.

Test case id	Test case description	Input	Expected Output	Test case result
01	The test case is dedicated to check the ability of the system to find out the hand of the person performing signs.	The input is the captured images of the hands for the system to distinguish.	The hand should be detected and indicated as shown in **fig. 7.5.**	Pass
02	To detect the hand signs used in the training model.	The images used for training the model are used to checked the results.	The expected output should be indicated as shown in **fig. 7.6.**	Pass
03	The images which are similar are tested to distinguish them.	The images of the signs such as “V” and “Y”, which are similar are used.	The system should indicate the correct letter for a specific sign with out any error.	Fail
04	The real-time captured images are used.	A real-time image is captured with the use of a webcam	The system should indicate the output of the image as shown in **fig. 7.7.**	Pass

Table 6.1. Test cases and their results

The test case one is mostly used to find out the ability of the proposed system to indentify the hand shown in the image. The result of this test case was mostly fruitful as there were only two classes to distinguish. The primary focus of the proposed system was on the test cases two and four. In the second test case we used the images that are already saved in the database to find out the result. The second test case mostly consists of images used for the development of the model. The result was satisfactory as shown in the fig. 7.6. for the letter “B”.

The third test case resulted in failure as shown in fig. 7.8. The system was not able to detect the letter “Y” when the hand was slightly crooked and wrongly identified it as letter “V”. The system also failed in detecting the hand signs which required motion. The images which are captured from the real-time feed is taken as the input for the final test case. For example, the below image shows the input and the result was shown in the fig. 7.7. for the letter “W”.

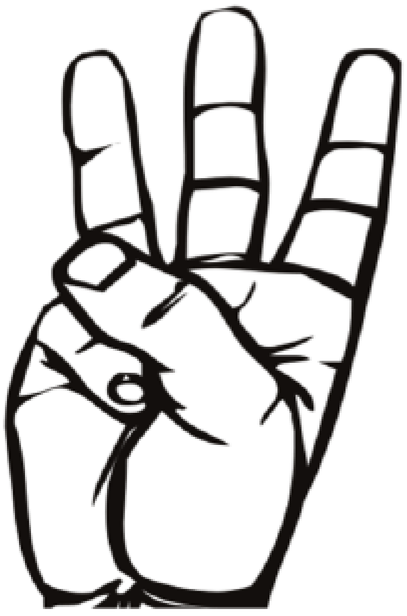
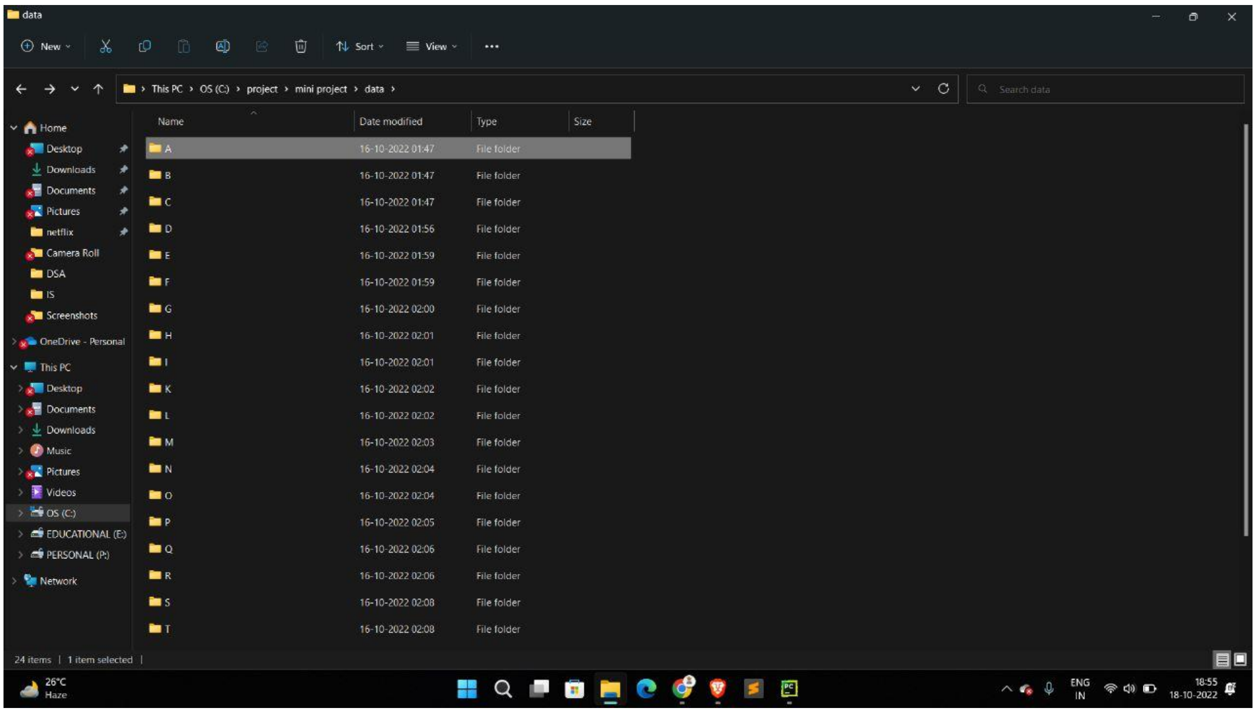


Fig 6.1. Letter “B” used as input

7. SCREEN SHOTS

Dataset:

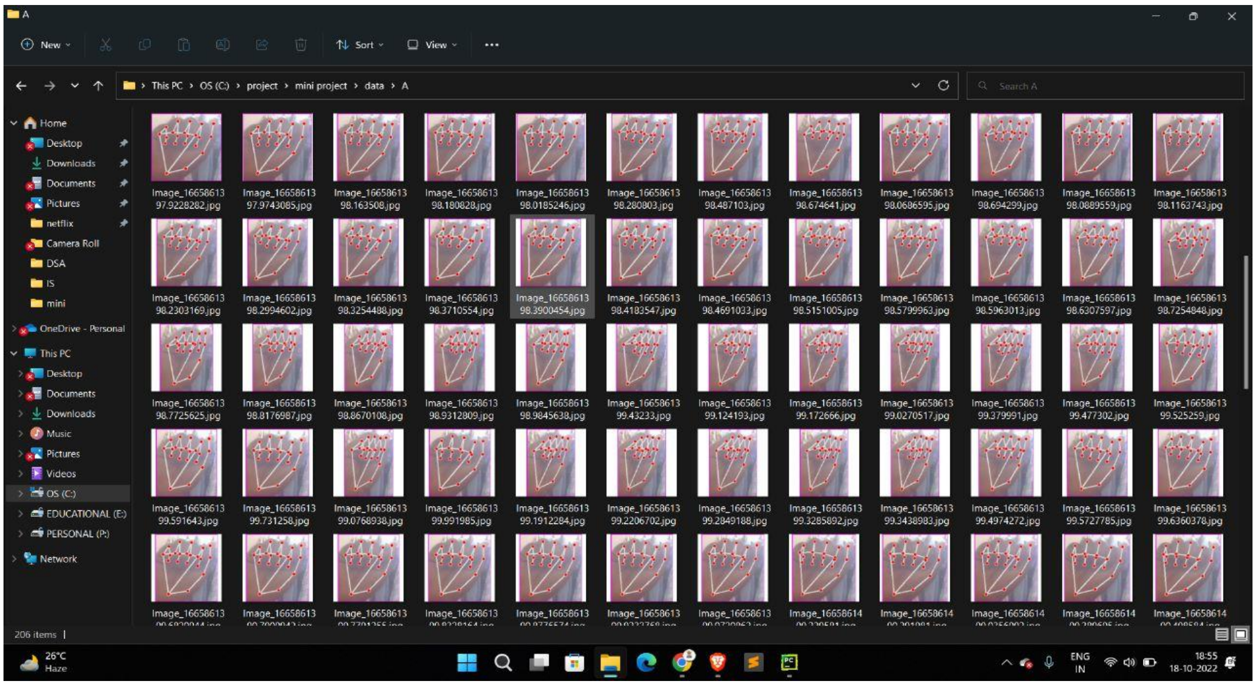


This PC › OS (C:) › project › mini project › data ›

Name	Date modified	Type	Size
A	16-10-2022 01:47	File folder	
B	16-10-2022 01:47	File folder	
C	16-10-2022 01:47	File folder	
D	16-10-2022 01:56	File folder	
E	16-10-2022 01:59	File folder	
F	16-10-2022 01:59	File folder	
G	16-10-2022 02:00	File folder	
H	16-10-2022 02:01	File folder	
I	16-10-2022 02:01	File folder	
K	16-10-2022 02:02	File folder	
L	16-10-2022 02:02	File folder	
M	16-10-2022 02:03	File folder	
N	16-10-2022 02:04	File folder	
O	16-10-2022 02:04	File folder	
P	16-10-2022 02:05	File folder	
Q	16-10-2022 02:06	File folder	
R	16-10-2022 02:06	File folder	
S	16-10-2022 02:08	File folder	
T	16-10-2022 02:08	File folder	

24 items | 1 item selected |

Fig 7.1. Data Folder



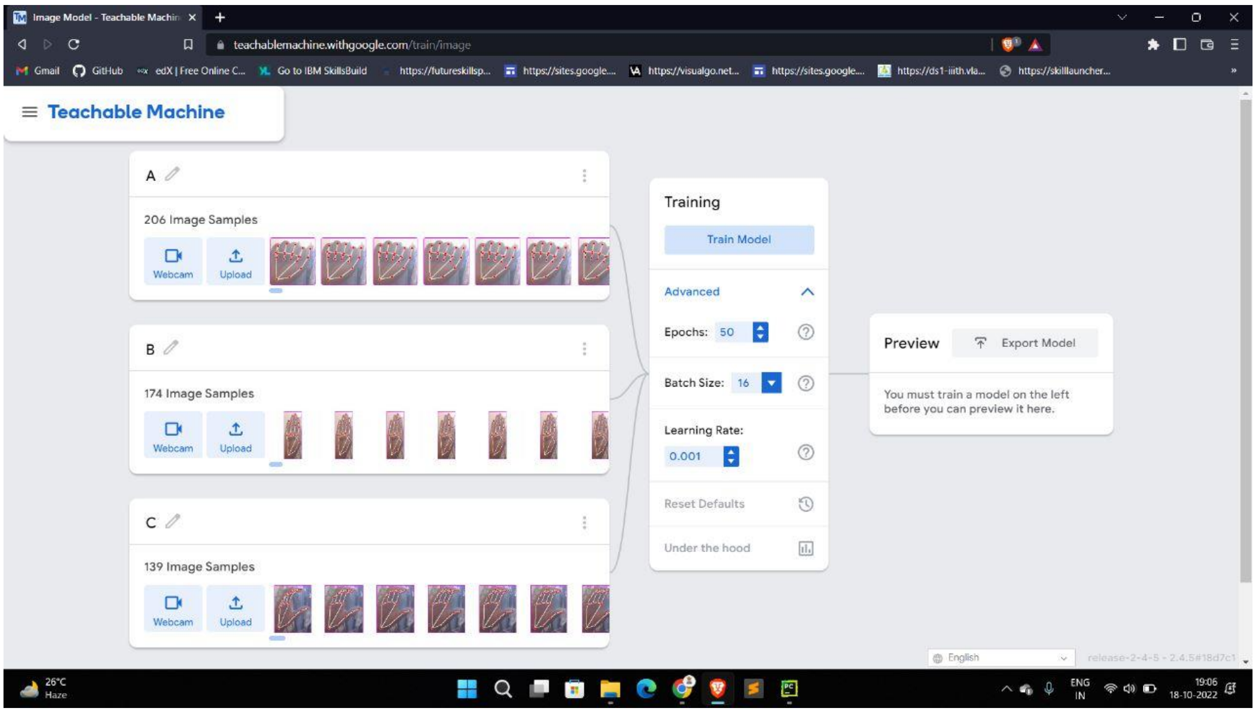
This PC › OS (C:) › project › mini project › data › A

Image_1665861397.9228282.jpg	Image_1665861397.9743085.jpg	Image_1665861398.163508.jpg	Image_1665861398.180828.jpg	Image_1665861398.0185246.jpg	Image_1665861398.280803.jpg	Image_1665861398.487103.jpg	Image_1665861398.674641.jpg	Image_1665861398.0686595.jpg	Image_1665861398.694299.jpg	Image_1665861398.0889559.jpg	Image_1665861398.1163743.jpg
Image_1665861398.2303169.jpg	Image_1665861398.2994602.jpg	Image_1665861398.3254488.jpg	Image_1665861398.3710554.jpg	Image_1665861398.3900454.jpg	Image_1665861398.4183547.jpg	Image_1665861398.4691033.jpg	Image_1665861398.5151005.jpg	Image_1665861398.5799963.jpg	Image_1665861398.5963013.jpg	Image_1665861398.6307597.jpg	Image_1665861398.7254848.jpg
Image_1665861398.7725625.jpg	Image_1665861398.8176987.jpg	Image_1665861398.8670108.jpg	Image_1665861398.9312809.jpg	Image_1665861398.9845638.jpg	Image_1665861399.43233.jpg	Image_1665861399.124193.jpg	Image_1665861399.172666.jpg	Image_1665861399.0270517.jpg	Image_1665861399.379991.jpg	Image_1665861399.477302.jpg	Image_1665861399.525259.jpg
Image_1665861399.591643.jpg	Image_1665861399.731258.jpg	Image_1665861399.0768938.jpg	Image_1665861399.991985.jpg	Image_1665861399.1912284.jpg	Image_1665861399.2206702.jpg	Image_1665861399.2849188.jpg	Image_1665861399.3285892.jpg	Image_1665861399.3438983.jpg	Image_1665861399.4974272.jpg	Image_1665861399.5727785.jpg	Image_1665861399.6360378.jpg
Image_16658613	Image_16658613	Image_16658613	Image_16658613	Image_16658613	Image_16658613	Image_16658613	Image_16658614	Image_16658614	Image_16658614	Image_16658614	Image_16658614

206 items |

Fig 7.2. Images Stored in the Database

Training Model:



A
206 Image Samples

B
174 Image Samples

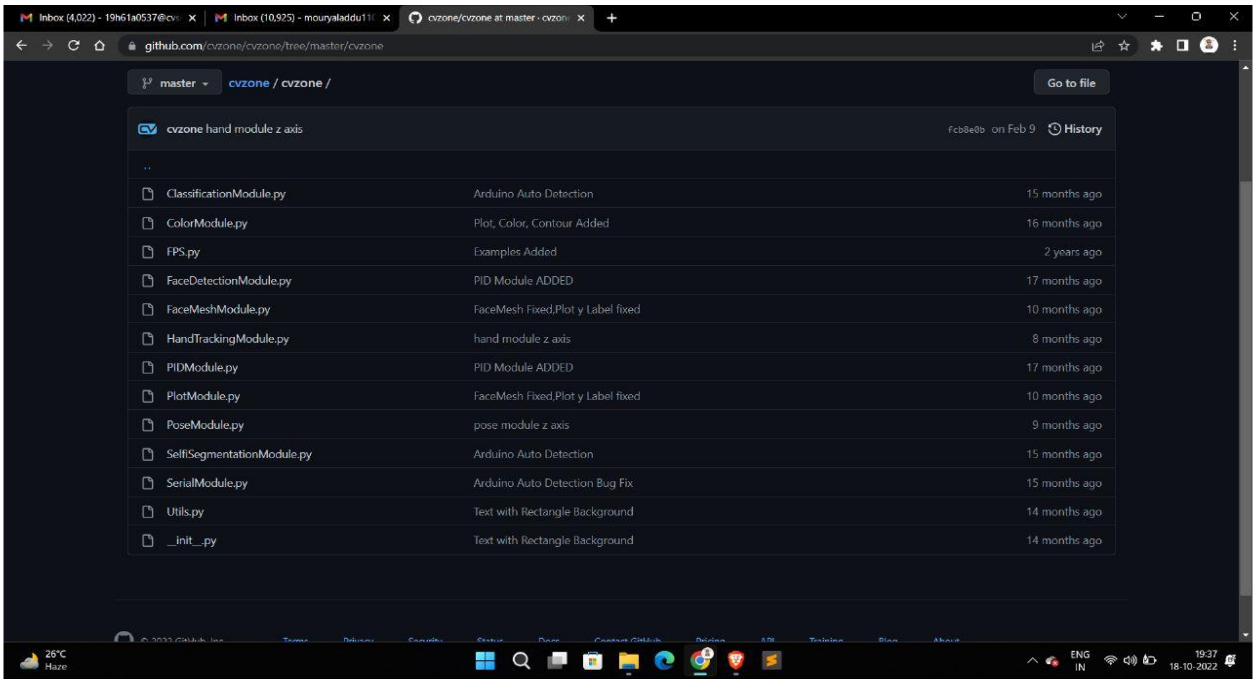
C
139 Image Samples

Epochs: 50
Batch Size: 16
Learning Rate: 0.001

You must train a model on the left before you can preview it here.

Fig 7.3. Training of the Model

Module:

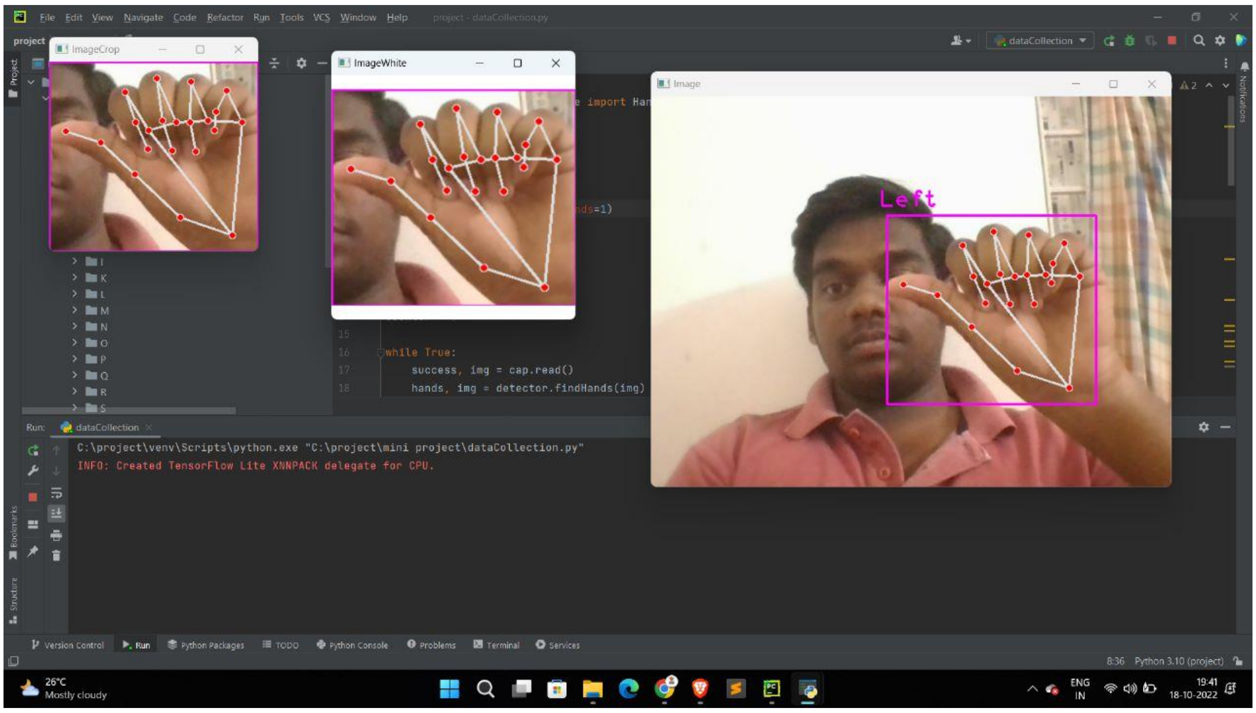


cvzone hand module z axis fcb8e0b on Feb 9

ClassificationModule.py	Arduino Auto Detection	15 months ago
ColorModule.py	Plot, Color, Contour Added	16 months ago
FPS.py	Examples Added	2 years ago
FaceDetectionModule.py	PID Module ADDED	17 months ago
FaceMeshModule.py	FaceMesh Fixed,Plot y Label fixed	10 months ago
HandTrackingModule.py	hand module z axis	8 months ago
PIDModule.py	PID Module ADDED	17 months ago
PlotModule.py	FaceMesh Fixed,Plot y Label fixed	10 months ago
PoseModule.py	pose module z axis	9 months ago
SelfiSegmentationModule.py	Arduino Auto Detection	15 months ago
SerialModule.py	Arduino Auto Detection Bug Fix	15 months ago
Utils.py	Text with Rectangle Background	14 months ago
__init__.py	Text with Rectangle Background	14 months ago

Fig 7.4. cvzone module

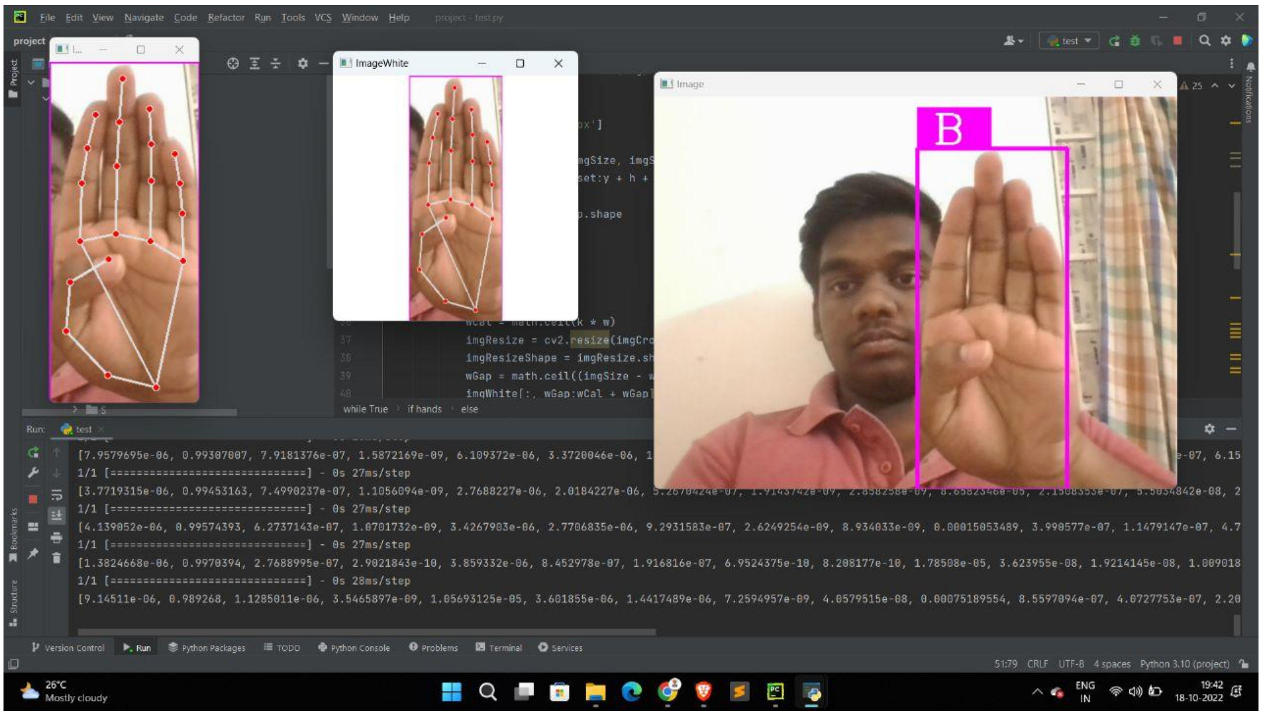
Training:



```
while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
```

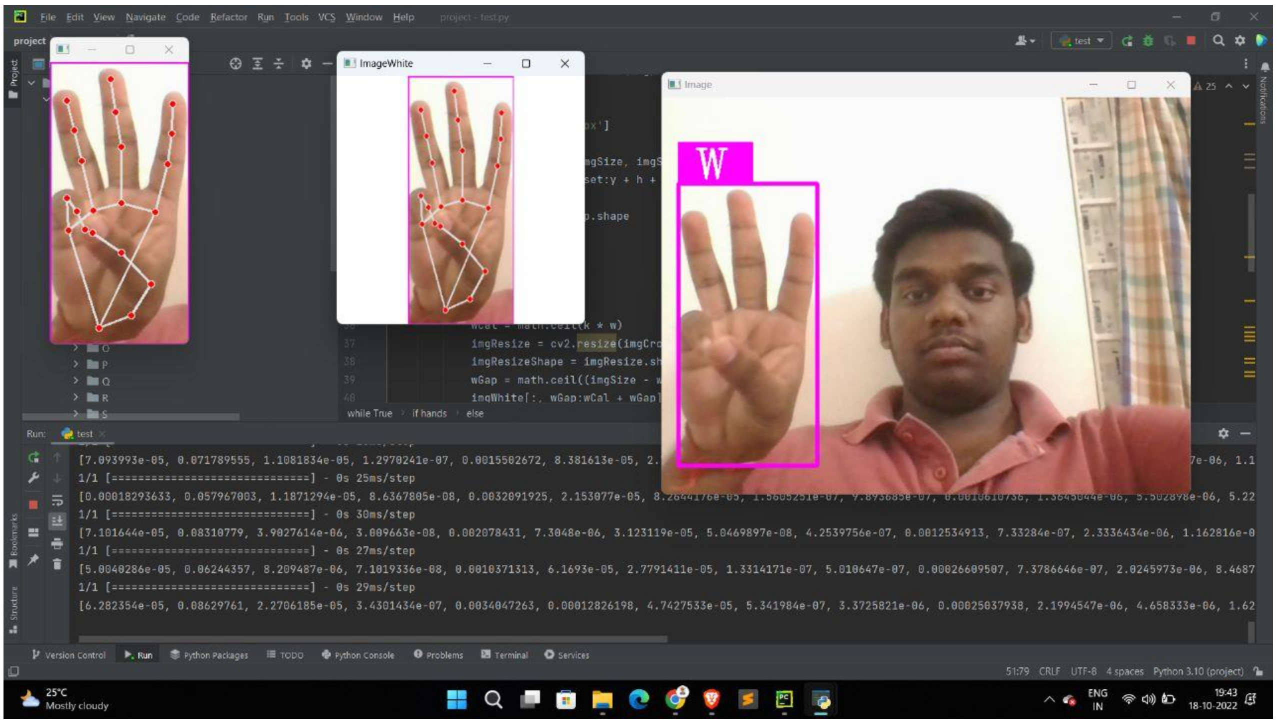
```
C:\project\venv\Scripts\python.exe "C:\project\mini project\dataCollection.py"
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Fig 7.5. Hand Detection



```
[7.9579695e-06, 0.99307007, 7.9181376e-07, 1.5872169e-09, 6.109372e-06, 3.3720046e-06, 1
1/1 [==============================] - 0s 27ms/step
[3.7719315e-06, 0.99453163, 7.4990237e-07, 1.1056094e-09, 2.7688227e-06, 2.0184227e-06, 5.2670424e-07, 1.9143742e-09, 2.8582586e-09, 8.6582346e-05, 2.1508355e-07, 5.5034842e-08, 2
1/1 [==============================] - 0s 27ms/step
[4.139052e-06, 0.99574393, 6.2737143e-07, 1.0701732e-09, 3.4267903e-06, 2.7706835e-06, 9.2931583e-07, 2.6249254e-09, 8.934033e-09, 0.00015053489, 3.990577e-07, 1.1479147e-07, 4.7
1/1 [==============================] - 0s 27ms/step
[1.3824668e-06, 0.9970394, 2.7688995e-07, 2.9021843e-10, 3.859332e-06, 8.452978e-07, 1.916816e-07, 6.9524375e-10, 8.208177e-10, 1.78508e-05, 3.623955e-08, 1.9214145e-08, 1.009018
1/1 [==============================] - 0s 28ms/step
[9.14511e-06, 0.989268, 1.1285011e-06, 3.5465897e-09, 1.05693125e-05, 3.601855e-06, 1.4417489e-06, 7.2594957e-09, 4.0579515e-08, 0.00075189554, 8.5597094e-07, 4.0727753e-07, 2.20
```

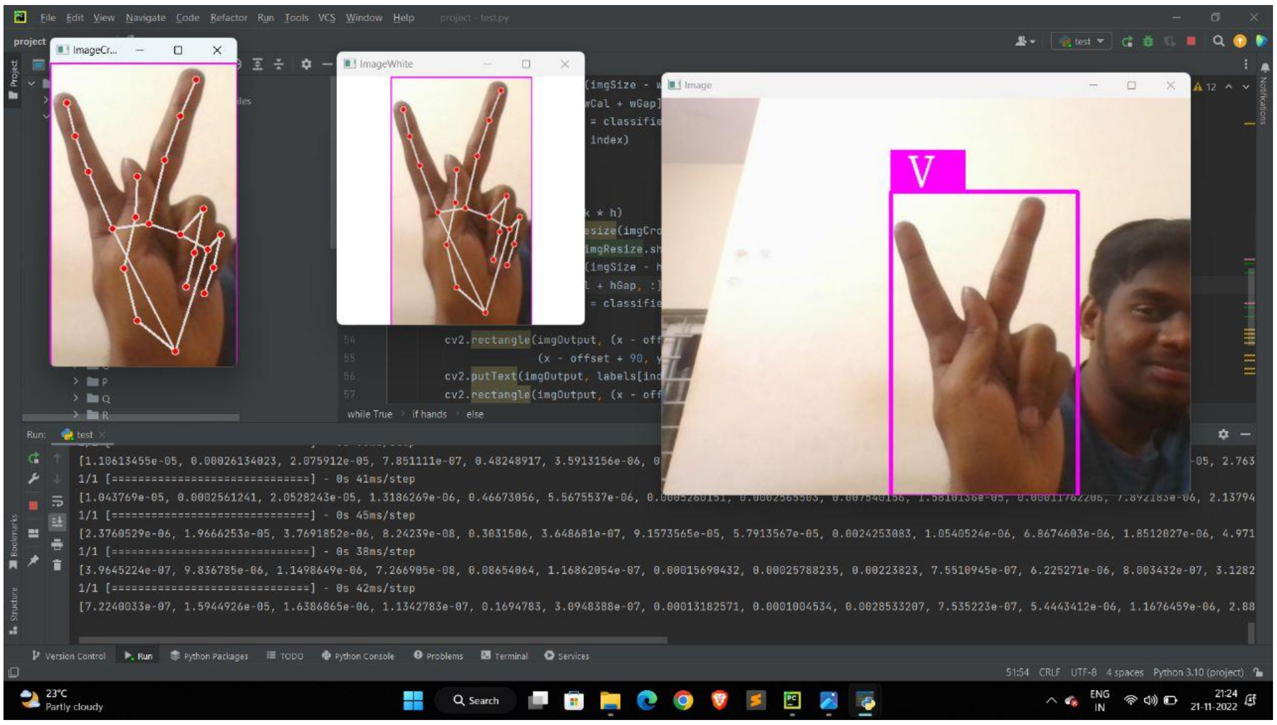
Fig 7.6. Successful result [i]



```
[7.093993e-05, 0.071789555, 1.1081834e-05, 1.2970241e-07, 0.0015502672, 8.381613e-05, 2.
1/1 [==============================] - 0s 25ms/step
[0.00018293633, 0.057967003, 1.1871294e-05, 8.6367805e-08, 0.0032091925, 2.153077e-05, 8.
1/1 [==============================] - 0s 30ms/step
[7.101644e-05, 0.08310779, 3.9027614e-06, 3.009663e-08, 0.002078431, 7.3048e-06, 3.123119e-05, 5.0469897e-08, 4.2539756e-07, 0.0012534913, 7.33284e-07, 2.3336434e-06, 1.162816e-0
1/1 [==============================] - 0s 27ms/step
[5.0040286e-05, 0.06244357, 8.209487e-06, 7.1019336e-08, 0.0010371313, 6.1693e-05, 2.7791411e-05, 1.3314171e-07, 5.010647e-07, 0.00026609507, 7.3786646e-07, 2.0245973e-06, 8.4687
1/1 [==============================] - 0s 29ms/step
[6.282354e-05, 0.08629761, 2.2706185e-05, 3.4301434e-07, 0.0034047263, 0.00012826198, 4.7427533e-05, 5.341984e-07, 3.3725821e-06, 0.00025037938, 2.1994547e-06, 4.658333e-06, 1.62
```

Fig 7.7. Successful result [ii]

Failed Test case:



```
[1.10613455e-05, 0.00026134023, 2.075912e-05, 7.851111e-07, 0.48248917, 3.5913156e-06, 0
1/1 [==============================] - 0s 41ms/step
[1.043769e-05, 0.0002561241, 2.0528243e-05, 1.3186269e-06, 0.46673056, 5.5675537e-06, 0.
1/1 [==============================] - 0s 45ms/step
[2.3760529e-06, 1.9666253e-05, 3.7691852e-06, 8.24239e-08, 0.3031506, 3.648681e-07, 9.1573565e-05, 5.7913567e-05, 0.0024253083, 1.0540524e-06, 6.8674603e-06, 1.8512027e-06, 4.971
1/1 [==============================] - 0s 38ms/step
[3.9645224e-07, 9.836785e-06, 1.1498649e-06, 7.266905e-08, 0.08654064, 1.16862054e-07, 0.00015690432, 0.00025788235, 0.00223823, 7.5510945e-07, 6.225271e-06, 8.003432e-07, 3.1282
1/1 [==============================] - 0s 42ms/step
[7.2240033e-07, 1.5944926e-05, 1.6386865e-06, 1.1342783e-07, 0.1694783, 3.0948388e-07, 0.00013182571, 0.0001004534, 0.0028533207, 7.535223e-07, 5.4443412e-06, 1.1676459e-06, 2.88
```

Fig 7.8. Failure Case

8. CONCLUSION

Nowadays, applications need several kinds of images as sources of information for analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore, the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image. Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using various methods to make the image more readable by the computer. Sign language recognition system is a powerful tool to prepare an expert knowledge, edge detect and the combination of inaccurate information from different sources. The intend of convolution neural network is to get the appropriate classification. In this project we faced many limitations like this project was unable to predict dynamic signs such as signs for letters “J” and “Z”. The prediction of more advanced signs which include the location of the hand and position of the palm is not possible. The image had to be captured and saved to predict the gesture. But inspite of these limitations we were able to achieve an accuracy of 94%. We want tackle these limitations in the future enhancement of the project.

9. FUTURE ENHANCEMENT

Now, the prediction is limited to images captured in real-time and can be enhanced further to predict the gestures from a live video in recording.
The extent of the signs predicted can be enhanced to even include more complex dynamic signs rather than the basic signs of ASL(American Sign Language).
The signs which include the location and palm position can also be included for a more pragmatic sign-to-text conversion.

10. BIBLIOGRAPHY

1. R. Harini, R. Janini, S. Keerthana, S. Madhubala, S. Venkatasubramanian, “Sign Lanaguage Translation”, DOI: 10.1109/ICACCS48705.2020.9074370, 2020.

2. Kshitij Bantupalli, Ying Xie, “American Sign Language Recognition using Deep Learning and Computer Vision”, DOI: 10.1109/BigData.2018.8622141, 2018.

3. Anup Kumar, Karun Thankachan, Mevin M. Dominic, “Sign Language Recognition”, DOI: 10.1109/RAIT.2016.7507939, 2016.

4. Niyol Roy,Tamanna Islam Juthi, Md AyobAnsari, “Real Time Dynamic Hand Gesture Recognition for Human Computer Interaction”, DOI: 10.1109/ROBIO.2015.7419071, 2018.

5. Mansour Alsulaiman, Mohamed A. Bencherif, and Mohamed Amine Mekhtiche, “Hand Gesture Recognition for Sign Language Using 3DCNN”, Citation information: DOI 10.1109/ACCESS.2020.2990434, IEEE Access,2017.

6. Purva C. Badhe, Vaishali Kulkarni, “Indian Sign Language Translator Using Gesture Recognition Algorithm”, DOI: 10.1109/CGVIS.2015.7449921, 2015.