A PROJECT REPORT ON

Sign Language Recognition System

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY,
PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING (Computer Engineering)

BY

Suraj Gawade	Exam No: B150314245
Saurabh Kumatkar	Exam No: B150314281
Geet Lakhe	Exam No: B150314284
Aditya Pise	Exam No: B150314312

Under the guidance of

Mr. Anand G. Deshmukh

DEPARTMENT OF COMPUTER ENGINEERING
PES MODERN COLLEGE OF ENGINEERING
SHIVAJINAGAR, PUNE 411005

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE 2021 - 22

CERTIFICATE

This is to certify that the Project Entitled

Sign Language Recognition System

Submitted by

Suraj Gawade	Exam No: B150314245
Saurabh Kumatkar	Exam No: B150314281
Geet Lakhe	Exam No: B150314284
Aditya Pise	Exam No: B150314312

is a bonafide work carried out by them under the supervision of Prof. Anand. G. Deshmukh and it is approved for the partial fulfilment of the requirement of Savitribai Phule Pune university, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering).

Mr. Anand. G. Deshmukh
Guide
Department of Computer Engineering

Prof. Dr. Mrs. S. A. ITKAR
Head
Department of Computer Engineering

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A Project Report Titled as

Sign Language Recognition System

is successfully completed by

Suraj Gawade	Exam No: B150314245
Saurabh Kumatkar	Exam No: B150314281
Geet Lakhe	Exam No: B150314284
Aditya Pise	Exam No: B150314312

DEPARTMENT OF COMPUTER ENGINEERING

PES MODERN COLLEGE OF ENGINEERING

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

ACADEMIC YEAR 2021-2022

Mr. Anand. G. Deshmukh
Guide
Department of Computer Engineering

Prof. Dr. Mrs. S. A. ITKAR
Head
Department of Computer Engineering

Acknowledgement

It gives us pleasure in presenting the project report on **'Sign Language Recognition System'**.

Firstly, we would like to express our indebtedness appreciation to our guide **Mr. Anand. G. Deshmukh.** His constant guidance and advice played very important role in successful completion of the project. He always gave us his suggestions, that were crucial in making this report as flawless as possible.

We would like to express our gratitude towards **Prof. Dr. Mrs. S. A. Itkar** Head of Computer Engineering Department, PES Modern College of Engineering for her kind co-operation and encouragement which helped us during the completion of this report.

Also, we wish to thank our Principal, **Prof. Dr. Mrs. K. R. Joshi** and all faculty members for their whole-hearted co-operation for completion of this report. We also thank our laboratory assistants for their valuable help in laboratory.

Last but not the least, the backbone of our success and confidence lies solely on blessings of dear parents and lovely friends.

Suraj Gawade
Saurabh Kumatkar
Geet Lakhe
Aditya Pise

Abstract

Communication is integral part of development of any society, but some communities of disabled people like hearing impaired, deaf, dumb are left behind in society. A system which can recognize hand poses and gestures of American Sign Language in real time, it will try to connect deaf and dumb people with rest of people. Existing solutions are slow, less accurate and require accessories like sensor gloves.

Sign language is one of the oldest and most natural form of language for communication, but since most people do not know sign language and interpreters are very difficult to come by, we have come up with a real time method using Convolutional Neural Network for fingerspelling based American Sign Language. In our method, the hand is first passed through a filter and after the filter is applied the hand is passed through a classifier which predicts the class of the hand gestures. This aims to form a communication bridge with the deaf, dumb peoples and rest of society.

Keywords

Machine Learning, Deep Learning, Convolutional Neural Network (CNN), Streamlit, WebRTC, Mediapipe, Streamlit Cloud, Precision, Accuracy, Data Analysis, OpenCV, TensorFlow.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Overview

Millions of people across the world live with the inability to speak, hear or sometimes both. To overcome the connection gap with deaf-mute individuals, sign language is utilized which is the most practiced literature in the deaf- mute community to interact with people and yield opinions. Deaf-mute is a phrase that is practiced historically to recognize an individual who is either deaf or both deaf and cannot speak as well, and in both circumstances, sign language is the method of communication for them. Sign language is a language that utilizes visual-manual approaches to communicate or convey meaning. Having a massive community deaf-mute people all over the world, sign language recognition has always fascinated researchers to develop sophisticated models that can successfully recognize sign languages. Because of not being a universal language, sign language differs in terms of languages and communities. Previously, various researches have been conducted on different sign languages. There are different sign languages for different communities.

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face.

1.2 Motivation

Communication is integral part of development of any society, but some communities of disabled people like hearing impaired, deaf, dumb are left behind in society. Huge part of society comes under the class of disabled which includes Deaf and Dumb. Only few of their relatives or close friends understand sign language. They are left behind in society also the opportunities they get are less. A system which can recognize hand poses and gestures of Indian Sign Language in real time will try to connect deaf and dumb people with rest of people. Existing solutions are slow, less accurate and require accessories like sensor gloves.

We aim to study about sign language recognition system developed by researchers. The video captured by camera gets converted to image frames, but these images contain unnecessary details like colour which reduces efficiency of our processor so we need to pre-process them before feeding to our deep learning model.

1.3 Problem Definition and Objectives

1.3.1 Problem Definition

To Create Desktop Application / Web Application for translation of American Sign Language with the help of Deep Learning algorithm which can consistently identify and classify [A-Z] ASL characters. Identified characters are shown on screen in text format.

1.3.2 Objectives

- To create Web Application for real time translation of American Sign Language into Text.
- To develop Convolutional Neural Network for prediction of gestures performed by users into the camera.

1.4 Project Scope/Limitations

1.4.1 Project Scope

- People from deaf community find it difficult to communicate with society, which makes them to hire translator guy for their daily work. Our system can eliminate this need of translator.
- Translating the sign gestures into text and audio format are vital parts of our system.
- System would also be helpful in the work from home case, where it would be used in online video conferences so that person using it won't feel isolated from others.
- Our society works on fact that everyone gets their rights to present their opinions. In these terms, this system would be helpful for uplifting the community of deaf people.
- This system can also be helpful for the deaf community schools, as they can stay in touch with the usual education system as well as being organized with other schools.

1.4.2 Project Limitations

- User experience is not good if internet connectivity of user is weak.
- Model accuracy decrease if user does not perform gestures appropriately.
- Light and background conditions of user also affect model accuracy because model is trained on limited dataset.

1.5 Methodologies for Problem Solving

1.5.1 Deep Learning

1) Read the dataset.
2) Perform Pre-processing on the dataset.
   2.1) RGB image to Grayscale conversion.
   2.2) Applying Gaussian Blur filter.
   2.3) Using Adaptive Threshold for edge detection.
3) Split the dataset into training and testing data with a 70-30 split.
4) On the training dataset use techniques like filtering, pooling etc.
5) Then build a Deep Learning model using CNN.
6) The model that has been built will now be used for test purposes and accuracies will be noted.
7) The accuracies can be further improved by hyper parameter tuning.
8) Build a web application using Streamlit and deploy the application to the cloud.

1.5.2 Classification Techniques

1.5.2.1 SVM

Generally, Support Vector Machines (SVM) is considered to be a classification approach but it can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional

space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes. SVM also used in Object Detection and image classification.

What do we need to do to convert a CNN into an SVM image classifier?

So, to do image classification using SVM we need to apply 2 changes:

I] Apply loss = “hinge” for binary & “squared hinge” for multi class classification.
ii] Apply regularizer in the final output layer & apply activation = “linear” for binary & “SoftMax” for multiclass classification.

1.5.2.2 Random Forest

The Random Forest (RF) algorithm is a supervised classification algorithm. It builds upon the concept of decision trees presented in the last session. The RF relies on many self-learning decision trees (i.e. “Forest”). The idea behind using many decision trees (i.e. an ensemble) is that many base learners can come to one strong and robust decision compared to a single DT. Different from the manual (expert-based) definition of decision rules we defined last week, the RF uses self-learning decision trees. These trees automatically define rules at each node based on a training dataset.

CHAPTER 2

LITERATURE SURVEY

Title	Author	Publication	Methods Used	Remarks
Classification of Sign Language Characters by Applying a Deep Convolutional Neural Network	M. M. Hasan, A. Y. Srizon, A. Sayeed and M. A. M. Hasan,	2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), 2020, pp. 434-438, doi: 10.1109/ICAICT51780.2020.9333456.	Deep Convolutional Neural Network	Proposed CNN model had better accuracy than traditional Machine Learning Algorithms like Random Forest, Linear SVM, etc.
Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network,	P. Das, T. Ahmed and M. F. Ali	2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1762-1765, doi: 10.1109/TENSYMP50017.2020.9230772	Convolutional Neural Network	Dataset was made by capturing images of 5 persons. • Training precision and validation accuracies for proposed was better than older models.
Colour image edge detection method based on multiscale product using Gaussian function,	N. Ben Youssef, A. Bouzid and N. Ellouze,	2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 2016, pp. 228-232, doi: 10.1109/ATSIP.2016.7523073.	Edge detection algorithm like Sobel, canny, etc. • Gaussian smoothing function.	Proposed method uses Gaussian smoothing function and multiscale product detect edges of coloured images.
Peer to Peer Multimedia Real-Time Communication System based on WebRTC Technology	Zinah Tareq Nayyef 1, Sarah Faris Amer 1, Zena Hussain ; Department of Computer science, Dijlah University College	International Journal of Engineering & Technology, 7 (2.9) (2018) 125-13	Web Real Time Communication	a web peer to peer real time communication system that allows users to communicate with high-speed data transmission over the communication channel using WebRTC technology, HTML5, and use Node.js server address

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

Following are the assumptions with respect to running app successfully:

- User is connected to stable internet connection.
- User is present in place where brightness is medium to high.
- User has good quality active inbuilt or external video camera.
- User performs gestures approximately similar to what standard American Sign Language gestures are.

3.2 Functional Requirements

3.2.1 System Feature 1 (Displaying real-time video feed of user)

- Display of user actions on the screen in real-time.
- Button to turn on/off users camera feed as needed.
- Option to select from multiple camera devices.

3.2.2 System Feature 2 (Displaying output of a single gesture user is performing)

- Display output related to gesture user is performing currently.
- Display appropriate message in following situations:

  1] If user is not performing gesture appropriately.

  2] If user's hand is not present or partially present in the camera.

3.2.3 System Feature 3 (Displaying output of all gestures user has performed till now)

- Display outputs of all gestures user has performed till now.
- Display space if user has not performed gesture for definite time period.

3.3 External Interface Requirements

3.3.1 User Interfaces

There is single page in application. Following interfaces are provided in this page:

1] Images for all gestures of American Sign Language are shown on the page. This will help users to perform accurate gestures.

2] Frame for displaying users live video feed.

3] Option to select device from multiple available camera devices.

4] Two output fields are present. One displays text related to gesture user is currently performing. Another is for displaying all the outputs generated till now i.e. whole sentence.

5] Clear button to delete the character which is generated recently.

6] Reset button to delete complete sentence and start from new.

7] Button to turn on/off the camera.

3.3.2 Hardware Interfaces

- Inbuilt or external Web Camera

3.3.3 Software Interfaces

- Web Browser

3.3.4 Communication Interfaces

- Web Real Time Communication (WebRTC)

3.4 Non-functional Requirements

3.4.1 Performance Requirements

- 5 – 15 seconds of time is required to load the application successfully based on user's internet connection speed.
- 0.5-1 seconds of delay is present in gesture prediction based on internet connection speed.

3.4.2 Security Requirements

- No data about user is collected or shared with others.
- Only authorized users are allowed to make changes in the backend.

3.4.3 Software Quality Attributes

- **Confusion Matrix:**
  A confusion matrix is a way to express how many of a classifier's predictions were correct, and when incorrect, where the classifier got confused (hence the name!). In a confusion matrix, the rows represent the true labels and the columns represent predicted labels. Values on the diagonal represent the number (or percent, in a normalized confusion matrix) of times where the predicted label matches the true label. Values in the other cells represent instances where the classifier mislabeled an observation; the column tells us what the classifier predicted, and the row tells us what the right label was. This is a convenient way to spot areas where the model may need a little extra training.

- **Precision**:
  Precision is the number of correctly-identified members of a class divided by all the times the model predicted that class. In the case of ASL, the precision score would be the number of correctly identified ASL gestures divided by the total number of times the classifier predicted gestures rightly or wrongly.

- **Recall**:

  Recall is the number of members of a class that the classifier identified correctly divided by the total number of members in that class. For ASL, this would be the number of actual ASL that the classifier correctly identified as such.

- **F1 score:**

  F1 score is a little less intuitive because it combines precision and recall into one metric. If precision and recall are both high, F1 will be high, too. If they are both low, F1 will be low. If one is high and the other low, F1 will be low. F1 is a quick way to tell whether the classifier is actually good at identifying members of a class, or if it is finding shortcuts (e.g., just identifying everything as a member of a large class).

3.5 System Requirements

3.5.1 Software Requirements

- Language: Python 3
- IDE : Google Colab
- Libraries:
  1] OpenCV-python
  2] Pandas
  3] streamlit
  4] NumPy
  5] streamlit-webrtc
  6] mediapipe
  7] tensorflow

3.5.2 Hardware Requirements

- Processor: 3.3 gigahertz (GHz) or faster processor or SoC
- RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit.
- Hard disk space: 16 GB for 32-bit OS 20 GB for 64-bit OS

3.6 Analysis Models: SDLC Model to be applied

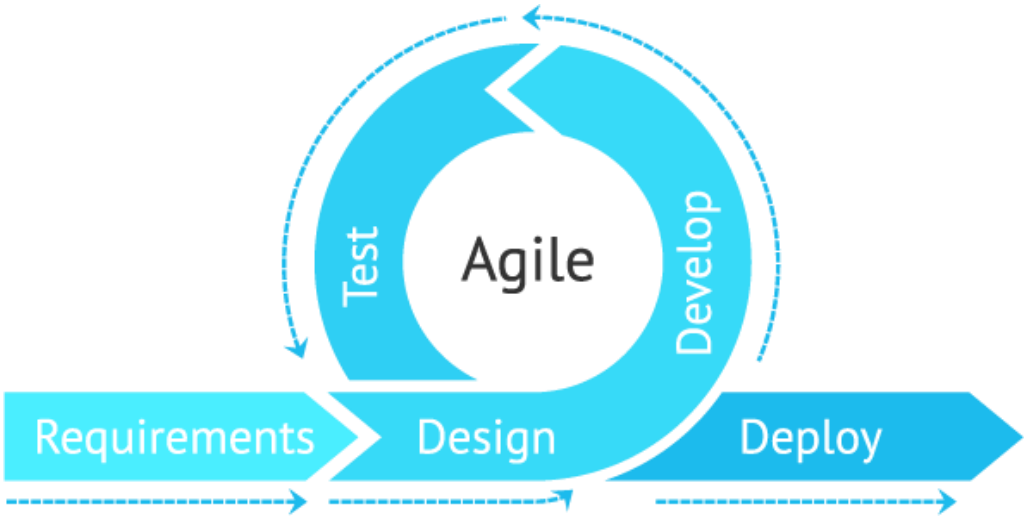


Figure 3.1 Agile Methodology

Agile software development refers to software development methodologies centered around the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

1] Requirements:

All required features of application were pointed out. What kind of dataset to be used was decided. Feasibility of all features was considered.

2] Design:

The second step was project design. The project was designed based on a framework. The framework uses three layers: – Business entities layer: It identifies all the entities used in the project. – Business logic layer: This layer operates on the business entity to achieve the goals. – Data access layer: This layer serves as an interface between back-end and the service. Custom dataset was created to be used to train CNN model.

3] Develop:

All modules and user interface were built in this step. Development was done using Python.

4] Test:

After implementation of each feature, complete application was tested. Testing was done for each new feature that was integrated into the application.

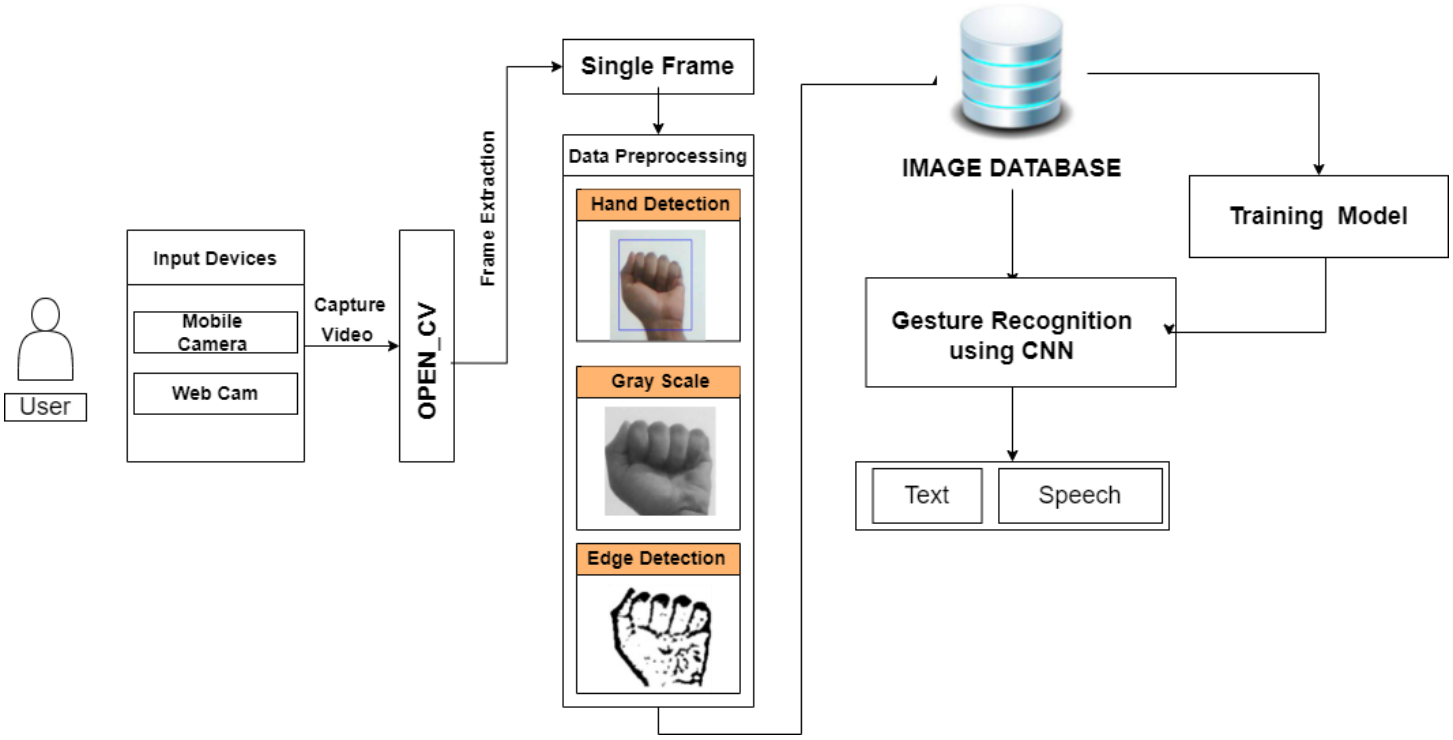
5] Deployment:

After all features were implemented and tested successfully, application was deployed on streamlit cloud.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture



4.2 Mathematical Model

1] Activation function:

A) <u>Relu:</u>

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input,but for any positive value x it returns that value back.

So it can be written as **f(x)=max(0,x) .**

B-SoftMax:

The SoftMax activation function is commonly used as an activation function in the case of multi-class classification problems in machine learning. The output of the SoftMax is interpreted as the probability of getting each class.

$$softmax(Z_i) = \frac{\exp(Z_i)}{\sum \exp(Z_i)}$$

2] Optimizer:

Adam:

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

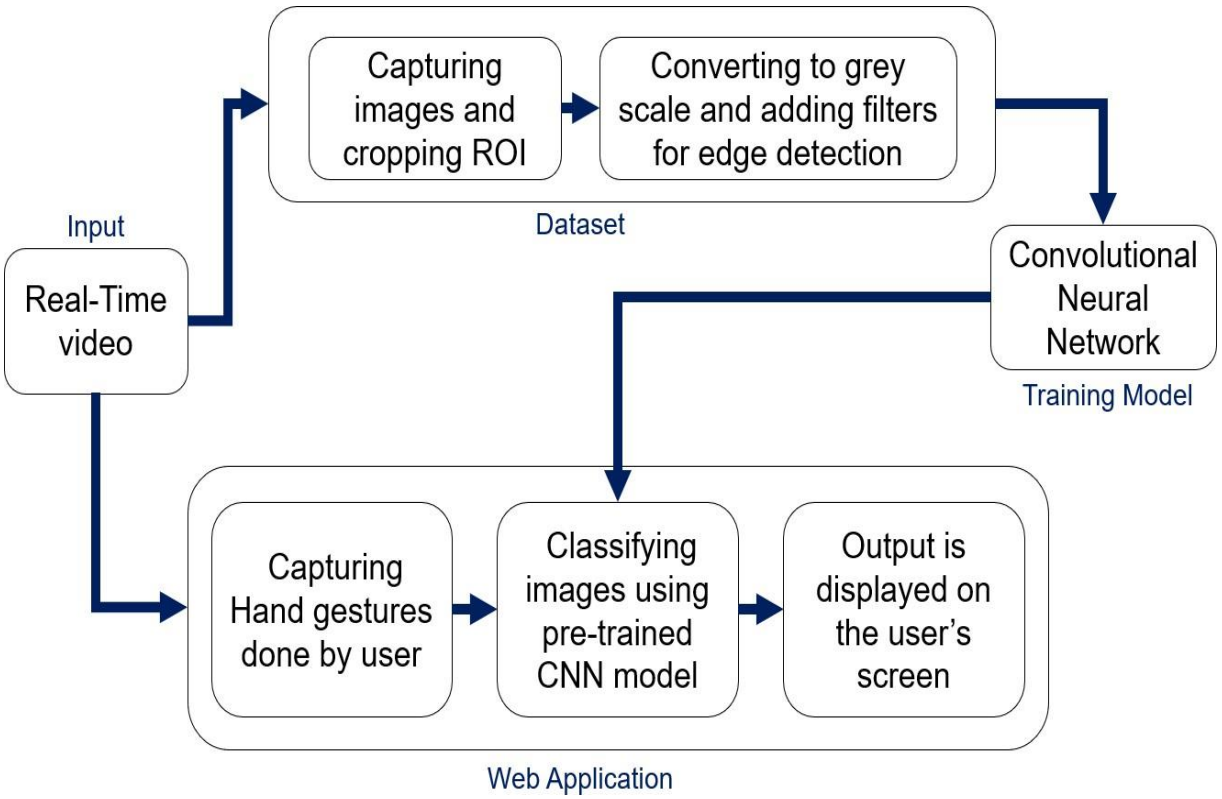
Mathematical Aspect of Adam Optimizer:

Taking the formulas used in the above two methods, we get

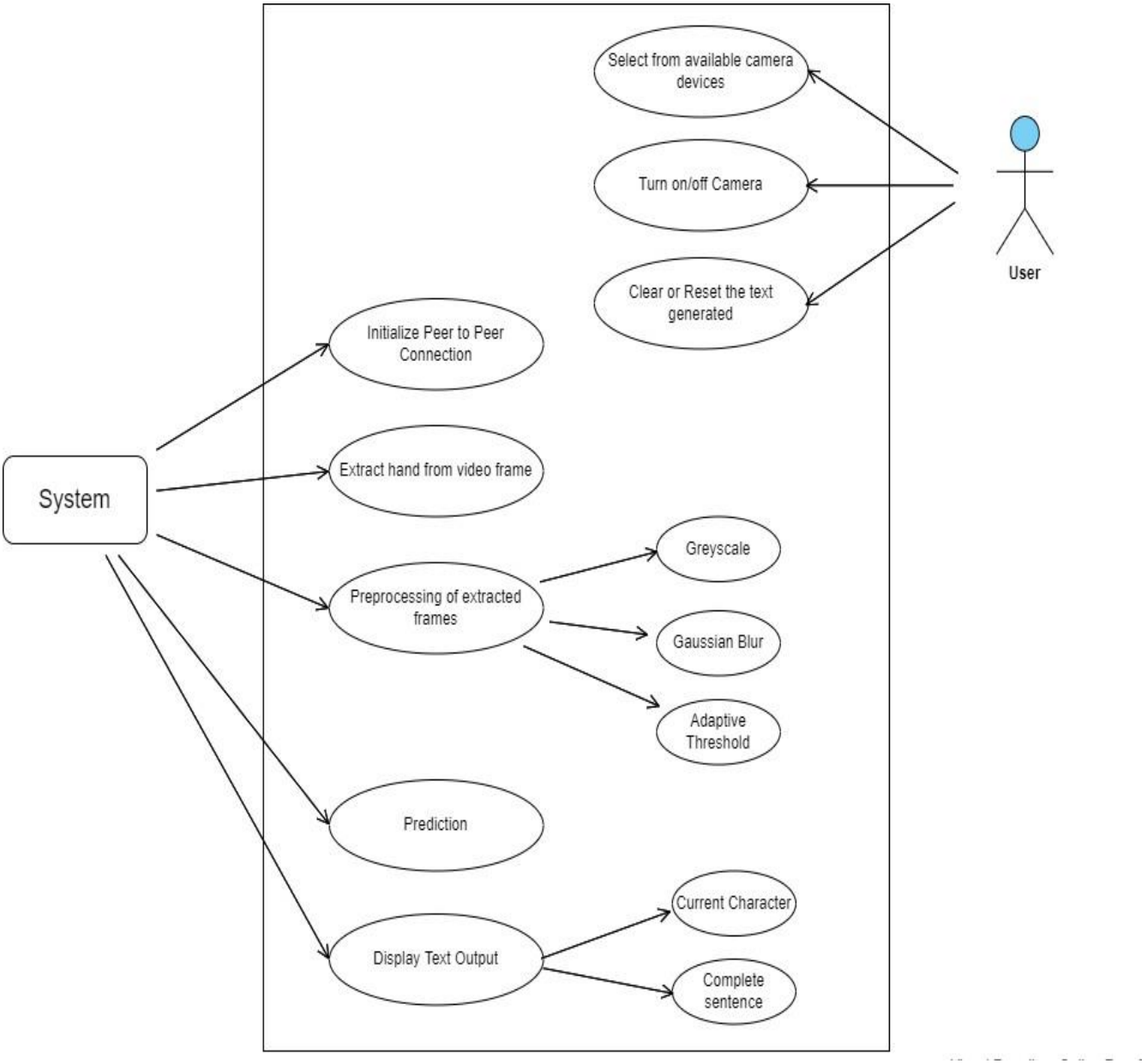
$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2$$

4.3 Data Flow Diagrams / UML Diagrams

4.3.1 Data Flow Diagram



4.3.2 Use Case Diagram



CHAPTER 5

PROJECT PLAN

5.1 Project Estimate

Analysis of available data to predict the time, cost, and resources needed to complete a project.

5.1.1 Reconciled Estimates

Cost: The cost of the project is nearly 0 as everything is open source. The python implementation done on google colab was free of cost as well as deploying on the cloud server did not cost any money.

Time: The implementation of the project would require about 60-90 days. A further 15 days could be required for testing and deployment while another 15 days would be necessary for the overall documentation of the project

5.1.2 Project Resources

- The project would require a team of members proficient in python language as well as some familiarity in streamlit and WebRTC.
- The project also requires a system with all the hardware and the software specifications mentioned above.

5.2 Risk Management

5.2.1 Risk Identification

1. Weak Internet Connectivity
2. Poor accuracy
3. System Incompatibility
4. Non-real time performance
5. Server crash

5.2.2 Risk Analysis

1] **Weak Internet Connectivity**: If internet connection of user is weak, there might be some delay to form initial connection between user and server. Also, time taken to predict the output will be somewhat more, which will lead to non-real experience to user.

2] **Poor accuracy**: If the training data is not accurate and if the models are not tuned properly, it may lead to a poor accuracy and hence may affect performance of the model. Also following factors may affect the accuracy of the system:

- User does not perform gestures appropriately.
- Camera quality is very poor.
- Too much objects in the background.

3] **System Incompatibility**: The system may not work if it is not compatible with the underlying software and hardware functionalities. For example, if browser version of user is too old, it might cause trouble while running application.

4] **Non-real time performance**: If internet connection is not strong, performance of application will not be smooth.

5] **Server Crash**: If server crashes for unknown reasons, users would not be able to access the application through their browsers.

5.2.3 Overview of Risk Mitigation, Monitoring, Management

1. CNN model will be trained on more images. Images used for training will be in different lighting and background conditions. Once model learns to predict on images on medium quality, accuracy will be better even if user does not perform gestures completely correct. Model will be adaptive to good to better lighting conditions in images. Even if there are other object in users background, model's accuracy would be high.

2. If user's browser does not support application, we will prompt user to upgrade their browser version.
3. We are using Streamlit Cloud platform to host our system. So, there are less chances of server crash. Even if server crashes, uptime will be very less because servers on streamlit cloud are managed by professionals.

5.3 Project Schedule

5.3.1 Project Task Set

Major Tasks in the Project stages are:

1. Selection of Topic
2. Literature Survey
3. Applications and Objectives
4. Platform and Technology Selection
5. Custom dataset creation
6. Study of Algorithms
7. System Architecture
8. Implementation of the Algorithms and building CNN model.
9. Construction of GUI
10. Deployment of the Website on a Cloud Service
11. Testing
12. Poster Presentation
13. Preparation of the Final Report

5.3.3 Timeline Chart

Sr. No.	Task Description	Start Date	End Date
1	Selection of topic	19/07/2021	29/07/2021
2	Literature Survey	02/08/2021	15/08/2021
3	Applications and Objectives	23/08/2021	30/08/2021
4	Platform and Technology Selection	06/09/2021	10/09/2021
5	Custom dataset creation	13/09/2021	25/09/2021
6	Study of Algorithms	26/09/2021	30/09/2021
7	System Architecture	15/10/2021	20/10/2021
8	Implementation of the Algorithms and building CNN model.	10/01/2022	13/02/2022
9	Construction of GUI	25/02/2022	10/03/2022
10	Deployment of the Website on a Cloud Service	11/03/2022	12/03/2022
11	Testing	28/03/2022	30/03/2022
12	Poster Presentation	20/04/2022	20/04/2022
13	Preparation of the Final Report	22/04/2022	01/05/2022

5.4 Team Organization

5.4.1 Team Structure

Our team comprises of 4 members. The aim of our team was to divide the work equally so that each team member can combine together to produce a final outcome. Regular meetings were held between the team members and deadlines were set in order to achieve the final output

Suraj Gawade: Literature Survey, Dataset Creation, CNN Model Creation, Documentation

Saurabh Kumatkar: Literature Survey, Dataset Creation, Web Application Creation, Documentation

Geet Lakhe: Literature Survey, Dataset Creation, Web Application Creation, Documentation

Aditya Pise: Literature Survey, Dataset Creation, CNN Model creation, Documentation

5.4.2 Management reporting and communication

For project guidance and reporting, we held regular meeting with our guide. The doubts and the problems that arose during this phase were resolved through meeting between our team members and our guide. Regular updates were shared with the guide and any improvements and feedbacks was worked upon.

Guide Name: Mr. Anand Deshmukh

CHAPTER 6 PROJECT IMPLEMENTATION

6.1 Overview of Project Modules

- **Convolutional Neural Network for Hand Gesture Recognition:**
  CNN model was created and trained on custom dataset. Custom dataset was created for 26 ASL alphabets of all team members. Whenever user performs hand gesture in camera, portion of image which includes user's hand is found out using Mediapipe library, then pre-processing is done and passed on to the CNN model for prediction.

- **Python Script for dataset creation:**
  Python program was developed for creation of custom dataset. Dataset includes images of 26 ASL alphabets of all members.

- **Streamlit module:**
  This module contains all the code necessary to implement Front-end as well as Back-end of the system. Streamlit is an isomorphic framework which handles both front-end and back-end in single framework.

6.2 Tools and Technologies Used

6.2.1 Google Colab:

Google Colab is an online tool that we have used for the implementation of our machine learning models. It is an online notebook in which we have performed operations like data pre-processing, visualizations, implementation of machine learning models and comparing each of the models.

6.2.2 Streamlit:

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code.

Characteristics of Streamlit:

- **Development flow**

If the source code of the streamlit's python script changes the app shows whether to rerun the application or not in the top-right corner. You can also select the 'Always rerun' option to rerun always when the source script changes.

This makes our development flow much easier, every time you make some changes it'll reflect immediately in your web app. This loop between coding and viewing results live makes you work seamlessly with streamlit.

- **Data flow**

Streamlit allows you to write an app the same way you write a python code. The streamlit has a distinctive data flow, any time something changes in your code or anything needs to be updated on the screen, streamlit reruns your python script entirely from the top to the bottom. This happens when the user interacts with the widgets like a select box or drop-down box or when the source code is changed.
If you have some costly operations while rerunning your web app, like loading data from databases, you can use streamlit's st.cache method to cache those datasets, so that it loads faster.

- **Widgets**

There are several widgets available in streamlit, like st.selectbox, st.checkbox, st.slider, and etc.

- **Layout**

You can easily arrange your widgets or data seamlessly using the st.sidebar method. This method helps you to align data in the left panel sidebar. All you have to do is simply use st.sidebar.selectbox to display a selectbox in the left panel.

6.2.3 WebRTC (Web Real-Time Communication):

It is a technology that enables Web applications and sites to capture and optionally stream audio and/or video media, as well as to exchange arbitrary data between browsers without requiring an intermediary. The set of standards that comprise WebRTC makes it possible to share data and perform teleconferencing peer-to-peer, without requiring that the user install plug-ins or any other third-party software.

WebRTC works on three primary components to initiate a peer-to-peer interaction effectively. These components have independent and crucial roles to play in the WebRTC specification.

A) Media Stream

Media Stream is an API that provides a way to access the camera and microphone of the device. It controls the multimedia activities of the device over the data consumed. The Media stream looks after the information of the device concerning capturing and rendering media. Ideally, it supports audio and video data streaming through the devices.

B) Peer Connection

WebRTC has all been developed to establish a peer-to-peer connection through the web. RTC peer connection has the primary objective of creating direct communication without the aid of any intermediary connection. Peers can even acquire or consume the media, specifically the audio and the video, and also produce it.

C) Data Channel

RTC data channel helps to create a bi-directional transfer of arbitrary data between peers. This works on SCTP (Stream Control Transmission Protocol). To reduce congestion on the networks like UDP, a data channel is designed. It ensures reliable delivery of stream over the web.

The steps involved in establishing communication through WebRTC

A) Signaling

Signaling refers to setting up and controlling a communication session. The peers connecting to a real-time communication send their stream to the server, which the server encodes and delivers to the receiving peer. This communication can be bi-directional. The source encodes the stream and sends them in a suitable resolution to another peer.

B) Connecting

Connecting refers to securing a bi-directional communication between two peers. In WebRTC, communication happens to be in P2P connection, rather than the client-server connection. The connection is equally distributed among the two communicating agents through their transport addresses.

C) ICE servers

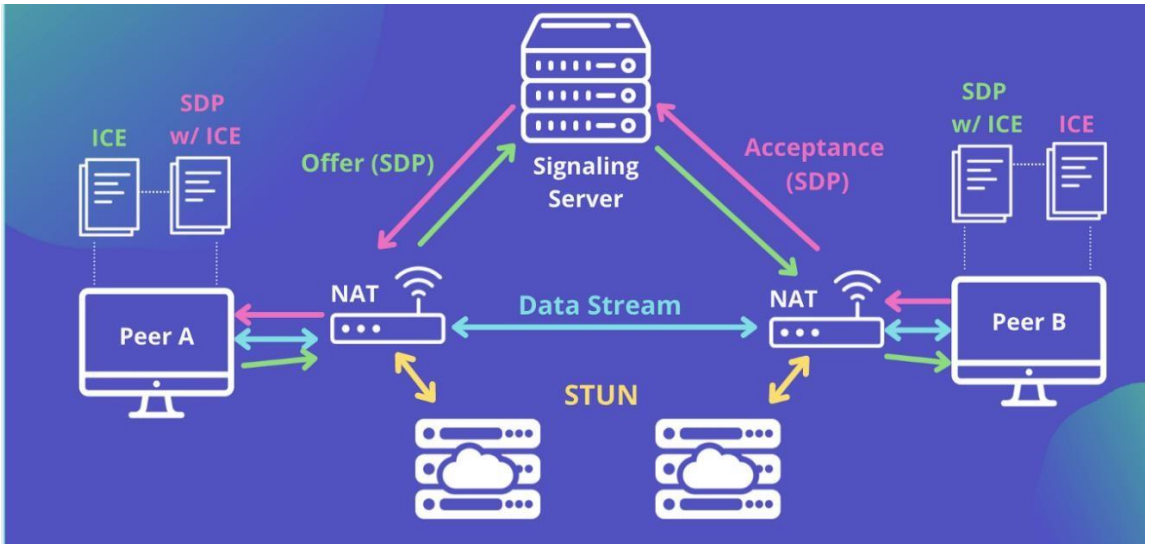
ICE is a protocol that tries to find out the best possible way to connect two agents or peers. Each ICE agent publishes its reachable known as candidates. These candidates ate nothing but the transport address of the agent to reach the other connecting agent.

D) STUN

STUN is an acronym for Standard Traversal Utilities for NAT. These lightweight servers allow WebRTC to find their public IP address by making Stun server requests.

E) TURN

TURN is an acronym for Traversal Using Relays around NAT. TURN servers help to establish connections between two agents when a direct connection between two agents is not possible due to firewall restrictions.



6.2.4 Mediapipe

Mediapipe is a pre-built python module used for machine learning purposes. Its specifically used to detect hands in a image or video.

6.2.5 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

6.3 Algorithm Details

6.3.1 Algorithm 1

- Apply Gaussian blur filter and threshold to the frame taken with opencv to get the processed image after feature extraction.
- This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and taken into consideration for forming the word.
- Space between the words is considered using the hand not found condition for 2sec.

6.3.2 Algorithm 2 (CNN)

1. 1st Convolution Layers Group:
The input picture has resolution of 64x64 pixels. It is first processed in the first convolutional layer using 64 filter weights (3x3 pixels each). This will result in a 62X62 pixel image, one for each Filter-weights. Another layer with 64 filter weights (3x3 pixels each) is used which will result in 60x60-pixel image.

2. 1st Pooling Layer:
The pictures are down sampled using max pooling of 2x2 i.e., we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 30x30 pixels.

3. 2nd Convolution Layers Group:
Now, these 30x30 from the output of the first pooling layer is served as an

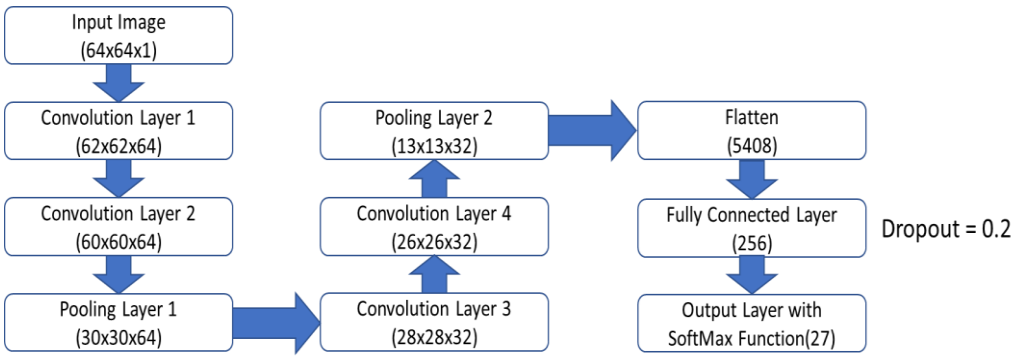
input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 28x28-pixel image. Another layer with same weights is added which results in 26x26 pixel image.

4. 2nd Pooling Layer:
The resulting images are down sampled again using max pool of 2x2 and is reduced to 13 resolutions of images.

5. Fully Connected Layer:
Now the output of last layer is flattened. Which results in one dimensional vector of 5408. The Dropout layer is added to randomly selecting nodes to be dropped-out with a 20% probability. Now this vector is used as an input to a dense layer with 256 neurons.

6. Final layer:
The output of the Fully Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

7. Batch Normalization Layer:
Batch Normalization layer is added after every convolutional layer to normalize the output of previous layer and pass it to the next layer.

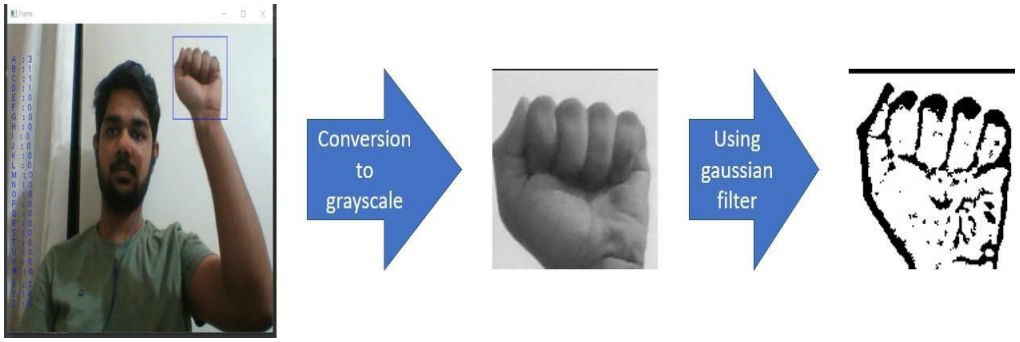


CNN architecture

6.3.3 Dataset Creation

- We have created our own custom dataset using python, opencv and media pipe.
- We captured and collected the images from group members.
- Then converted the rgb images to greyscale for better prediction.

ASL gestures



CHAPTER 7

SOFTWARE TESTING

7.1 Types of Testing

- **Unit testing:**
  Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

- **Integration testing:**
  Integration testing is used to check whether the small components accurately interact with each other accordingly to the instruction.

- **System testing:**
  System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

- **White Box testing:**
  White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

- **Regression testing:**
  Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of

already executed test cases which are re-executed to ensure existing functionalities work fine.

- **Load testing:**
  Load testing is a kind of Performance Testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

- **Stress testing:**
  Stress testing is the process of determining the ability of a computer, network, program or device to maintain a certain level of effectiveness under unfavorable conditions. The process can involve quantitative tests done in a lab, such as measuring the frequency of errors or system crashes.

- **Smoke testing:**
  Smoke testing is also known as 'Build Verification Testing', is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work. The result of this testing is used to decide if a build is stable enough to proceed with further testing.

- **Validation Testing:**
  Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfils its intended use when deployed on appropriate environment.

7.2 Test Cases Test Results

Sr No	Test Case	Objective	Steps	Expected Result	Actual Result
1	Document Load	To check if webpage load without any error	Run Web app URL	Home page load.	Home page load
2	Starting Camera	To check if camera start without any error	Click start button.	User real time video visible to user.	User real time video visible to user
3	WebRTC peer to peer connection	To check if connection between client and server is established successfully .	Click start button	Connection successfully established	Connection established successfully
4	Identifying correct performed hand gesture.	To check if CNN model predict correct ASL alphabet or not.	User perform correct hand gesture as per standard gesture	Correct ASL alphabet in result tab.	Correct ASL alphabet in result tab.

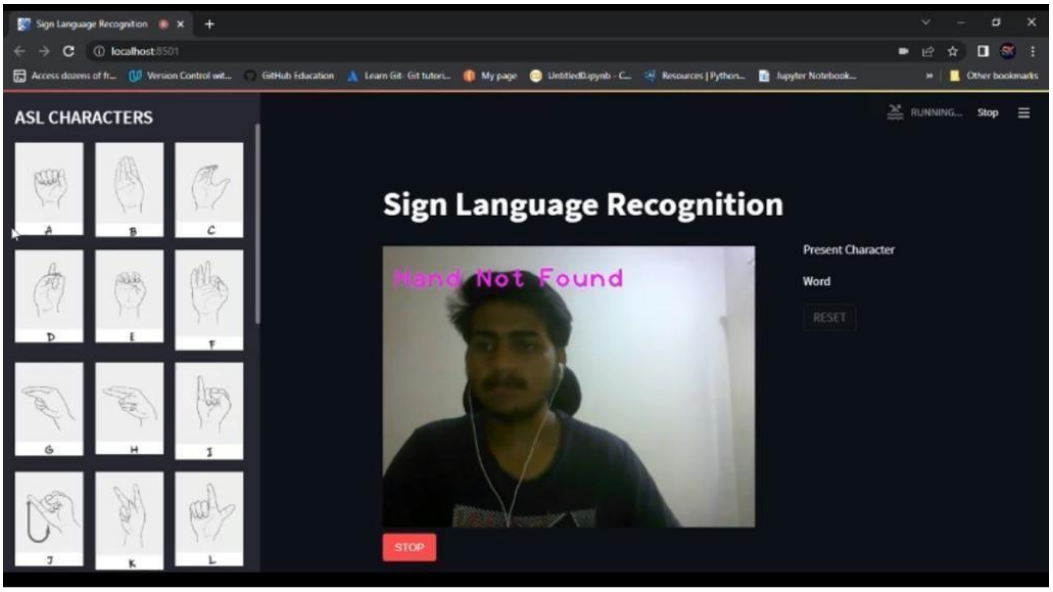
CHAPTER 8

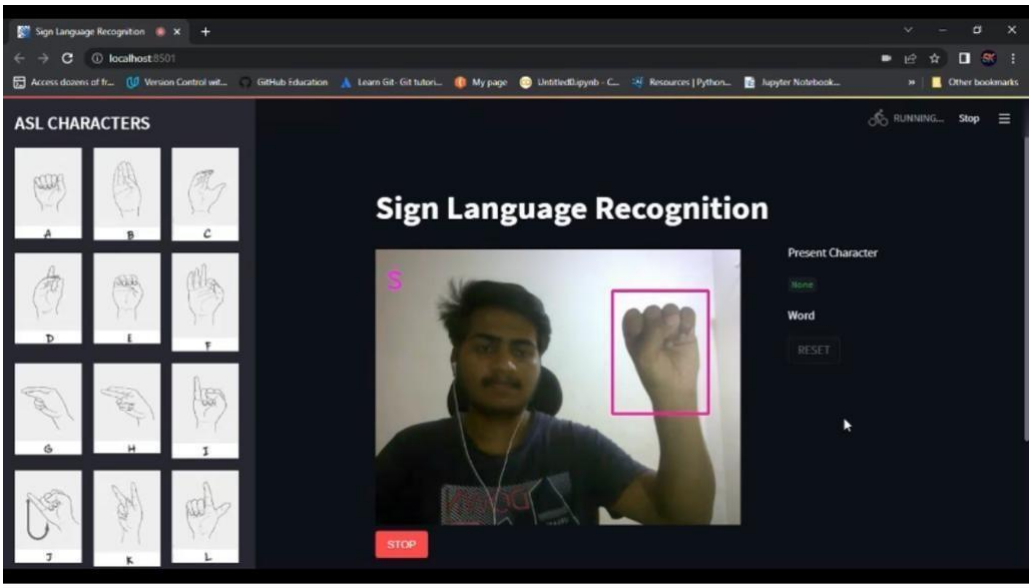
RESULTS

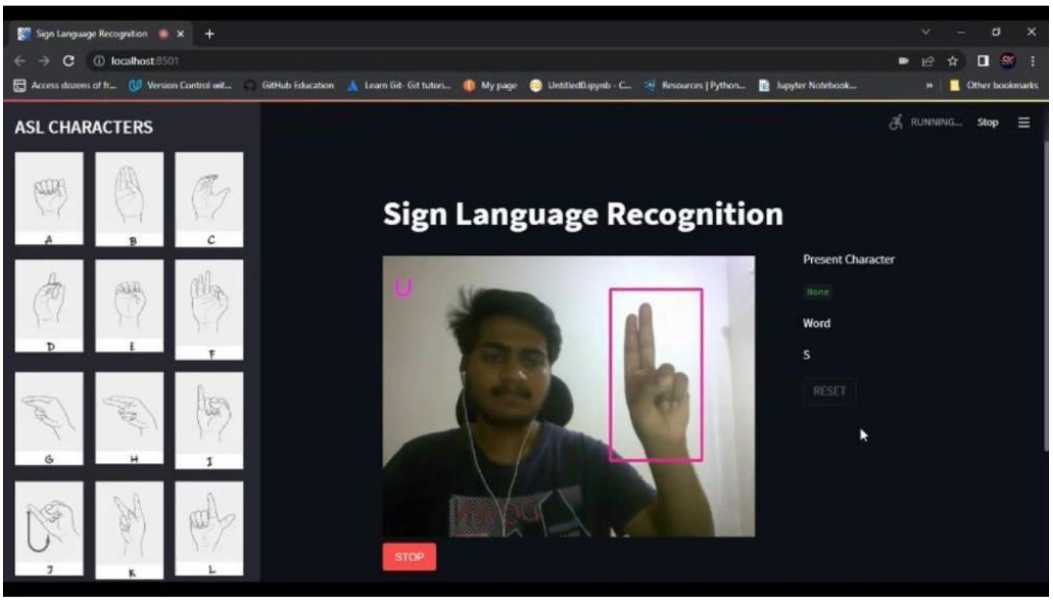
8.1 Outcomes

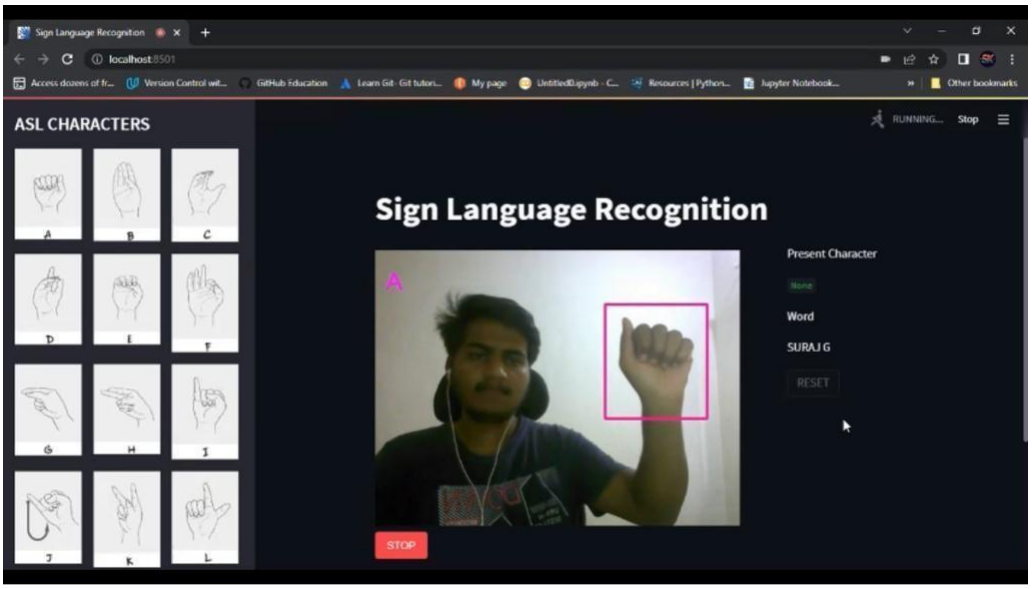
- Successful establishment of peer-to-peer connection between user's browser and server.
- Camera device accessed successfully.
- Images of user's hand transferred to server for prediction successfully.
- Accurate prediction of gestures.
- Text output displayed to user on browser successfully.

8.2 Screen Shots






Sign Language Recognition
localhost:8501
ASL CHARACTERS
RUNNING...
Stop
Sign Language Recognition
Present Character
None
Word
S
RESET
STOP


Sign Language Recognition
localhost:8501
ASL CHARACTERS
RUNNING...
Stop
Sign Language Recognition
Present Character
None
Word
SURAJ G
RESET
STOP

CHAPTER 9

CONCLUSIONS

9.1 Conclusions

- In this project, a functional real time vision based American Sign Language recognition for deaf and mute people will be developed for ASL alphabets.
- Convolutional Neural Networks will be used to increase the efficiency of the project.
- Input images for CNN model will be converted to grey scale for fast execution.
- Output will be given in both text and speech format.

9.2 Future Work

- We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms.
- We are also thinking of improving the pre-processing to predict gestures in low light conditions with a higher accuracy.

9.3 Applications

- In Video meeting applications, this system can be used for the deaf and Dumb people.
- In schools for D&D people, this system will make learning easier by allowing interaction between students and teachers.

ANNEXURE A

- **OpenCV –**

  OpenCV (Open-Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multicore processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

- **Convolution Neural network –**

  CNNs use a variation of multilayer perceptron's designed to require minimal pre-processing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

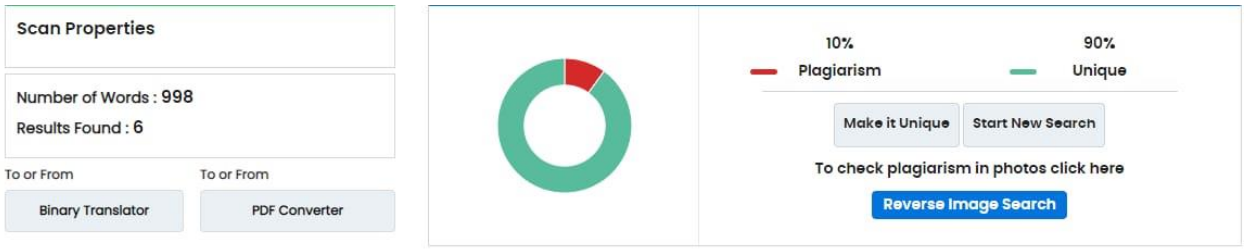
- **TensorFlow –**

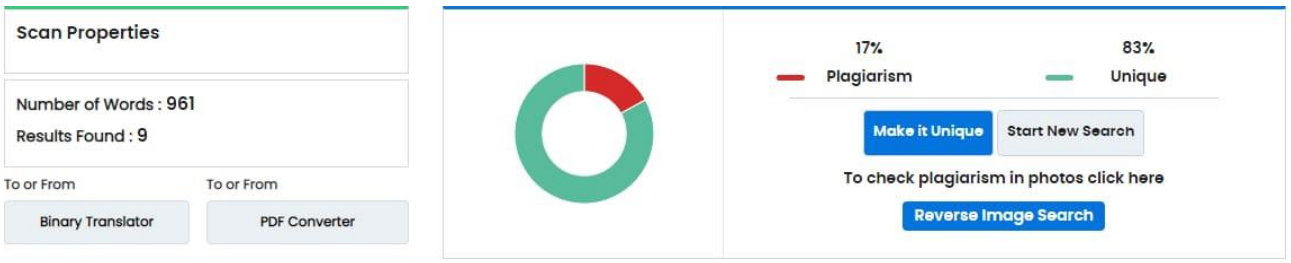
  is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library and is also used for

machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google brain team for internal Google use. It was released under the Apache 2.0 open-source library on November 9, 2015. TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

ANNEXURE B

Plagiarism Report:





CHAPTER 10
REFERENCES

[1] M. M. Hasan, A. Y. Srizon, A. Sayeed and M. A. M. Hasan, "Classification of Sign Language Characters by Applying a Deep Convolutional Neural Network," 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), 2020, pp. 434-438, doi: 10.1109/ICAICT51780.2020.9333456.

[2] W. Li, H. Pu and R. Wang, "Sign Language Recognition Based on Computer Vision," 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2021, pp. 919- 922, doi: 10.1109/ICAICA52286.2021.9498024.

[3] P. Das, T. Ahmed and M. F. Ali, "Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network," 2020 9 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1762- 1765, doi: 10.1109/TENSYMP50017.2020.9230772.

[4] N. Ben Youssef, A. Bouzid and N. Ellouze, "Color image edge detection method based on multiscale product using Gaussian function," 2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 2016, pp. 228-232, doi: 10.1109/ATSIP.2016.7523073.

[5] Thongtawee, O. Pinsanoh and Y. Kitjaidure, "A Novel Feature Extraction for American Sign Language Recognition Using Webcam," 2018 11th Biomedical Engineering International Conference (BMEiCON), 2018, pp. 1-5, DOI: 10.1109/BMEiCON.2018.8609933.

[6] https://en.wikipedia.org/wiki/American_Sign_Language

[7] https://www.mygreatlearning.com/blog/introduction-to-image-pre-processing

[8] https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

[9] 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)

[10] Home - OpenCV

[12] https://en.wikipedia.org/wiki/TensorFlow

[13] Convolutional neural network - Wikipedia