

**TRIBHUVAN UNIVERSITY**

**Institute of Science and Technology**

**A Project Report**

**On**

**“SignaLink:**  
**ASL Translation Using Feedforward Neural Network and Convolutional Neural Network with Ana­­lysis”**

**Submitted to:**

**Department of Computer Science and Information Technology**

**National College of Computer Studies**

*In partial fulfillment of the requirements for the Bachelor’s Degree in Computer*

*Science and Information Technology*

**Submitted By:**

**Abhijeet Yadav (23813/076)**

**Bibek Thakuri (23818/076)**

**Prajwal Shrestha (Malla) (23838/076)**

**Under the Supervision of**

**Chhetra Bahadur Chhetra**

**Date: 28th Falgun, 2080**

**NATIONAL COLLEGE OF COMPUTER STUDIES**

**TRIBHUVAN UNIVERSITY**

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project, prepared under my supervision entitled “SignaLink:ASL Translation Using Feedforward Neural Network and Convolutional Neural Network With Ana­­lysis”, a platform that detects the American Sign Language and translate into corresponding speech and vice versa, the system also provides user functionality to analyze the two different models, in partial fulfilment of the requirements for the degree of B.Sc. in Computer Science and Information Technology is processed for the evaluation.

……………………………….

**Mr. Chhetra Bahadur Chhetri**

Project Supervisor

National College of Computer Studies

Paknajol, Kathmandu

# CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “SignaLink:ASL Translation Using Feedforward Neural Network and Convolutional Neural Network With Ana­­lysis”, is a bonafide report of the work carried out by Mr. Abhijeet Yadav, Mr. Bibek Thakuri, Prajwal Shrestha (Malla) under the guidance and supervision for the degree of B.Sc. in Computer Science and Information Technology at National College of Computer Studies, Tribhuvan University.

To the best of my knowledge and belief, this work embodies the work of candidates, has duly been completed, fulfils the requirement of the ordinance relating to the Bachelor degree of the university and is up to the standard in respect of content, presentation, and language for being referred to the examiner.

|  |  |
| --- | --- |
| ………………………………..  **Mr. Chhetra Bahadur Chhetri**  Project Supervisor  National College of Computer Studies | ………………………………..  **Mr. Rajan Paudel**  Program Coordinator  National College of Computer Studies |
| ………………………………..  **External Examiner**  Tribhuvan University | ……………………………..  **Internal Examiner**  National College of Computer Studies |

# ACKNOWLEDGEMENT

The successful realization of our final year project owes much to the invaluable support extended by our project supervisors **Mr. Chhetra Bahadur Chhetri**, our designated supervisor, deserves profound appreciation and our sincere gratitude. We would also like to express our thanks to the National College of Computer Studies for providing us with an exceptional platform to pursue and develop this project.

Under the guidance of **Mr. Chhetra Bahadur Chhetri** and the NCCS team, our team has significantly deepened its understanding of AI and ML algorithms, as well as various components associated with the development of this system. This experience has equipped us with a comprehensive knowledge of the intricate workings behind complex AI systems, positioning us well for future real-life projects in this domain.

We extend our immense gratitude to the NCCS team for their thorough review, approval, and guidance throughout this transformative journey. Special acknowledgment is also due to the supportive online communities, as well as our friends and families, whose assistance was instrumental in the proper design, construction, and formation of this application.

In conclusion, we are truly honored by the collaborative efforts and support that have contributed to the success of our project.

**Abhijeet Yadav**

**Bibek Thakuri**

**Prajwal Shrestha (Malla)**

**Date: 26/08/2023**

# ABSTRACT

The sign language translation system aims to bridge the communication gap between individuals who use sign language and those who do not. This project presents a way to develop a system capable of translating sign language into text and vice versa. The project uses two algorithms Feedforward Neural Network (FNN) and Convolutional Neural Network (CNN) . The data set used for both algorithms is a custom dataset that mimics the MNIST dataset where each pixel of a 28\*28 grayscale image is represented between 0 to 255.Both algorithms use the dataset to train and recognize hand signs according to their respective labels. After opening the camera, when the user classifies the hand sign, the system displays the corresponding text and plays the corresponding audio. The test accuracy of CNN was found to be 94% and FNN was found to be 86%. The system also has a dashboard where the accuracy, precision , recall and f1 score can be visualized using different graphs and charts..

Keyword: Sign language translation system, FNN, CNN, MNIST dataset, Grayscale image

.

# TABLE OF CONTENTS

[SUPERVISOR’S RECOMMENDATION ii](#_Toc161012473)

[CERTIFICATE OF APPROVAL iii](#_Toc161012474)

[ACKNOWLEDGEMENT iv](#_Toc161012475)

[ABSTRACT v](#_Toc161012476)

[TABLE OF CONTENTS vi](#_Toc161012477)

[LIST OF ABBREVIATIONS viii](#_Toc161012478)

[LIST OF FIGURES ix](#_Toc161012479)

[LIST OF TABLES x](#_Toc161012480)

[Chapter 1: Introduction 1](#_Toc161012481)

[1.1 Introduction 1](#_Toc161012482)

[1.2 Problem Statement 1](#_Toc161012483)

[1.3 Objective 1](#_Toc161012484)

[1.4 Scope and Limitation 1](#_Toc161012485)

[1.5 Development Methodology 2](#_Toc161012486)

[1.6 Report Organization 3](#_Toc161012487)

[Chapter 2: Background Study and Literature Review 4](#_Toc161012488)

[2.1 Background Study 4](#_Toc161012489)

[2.2 Literature Review 4](#_Toc161012490)

[Chapter 3: Requirement Analysis and Feasibility Study 6](#_Toc161012491)

[3.1 System Analysis 6](#_Toc161012492)

[3.1.1 Requirement Analysis 6](#_Toc161012493)

[3.1.2 Feasibility Study 8](#_Toc161012494)

[3.1.3 Analysis 9](#_Toc161012495)

[Chapter 4: System Design 12](#_Toc161012496)

[4.1 Design 12](#_Toc161012497)

[4.1.1 Interface Design 17](#_Toc161012498)

[4.2 Algorithm Details 19](#_Toc161012499)

[Chapter 5: Implementation and Testing 24](#_Toc161012500)

[5.1 Implementation 24](#_Toc161012501)

[5.1.1 Tools Used 24](#_Toc161012502)

[5.1.2 Modules Description 26](#_Toc161012503)

[5.2 Testing 32](#_Toc161012504)

[5.2.1 Unit Testing 32](#_Toc161012505)

[5.2.2 System Testing 34](#_Toc161012506)

[5.3 Result Analysis 38](#_Toc161012507)

[5.3.1 Evaluating Accuracy 38](#_Toc161012508)

[Chapter 6: Conclusion and Future Improvements 43](#_Toc161012509)

[6.1 Conclusion 43](#_Toc161012510)

[6.2 Future improvement 43](#_Toc161012511)

[References 44](#_Toc161012512)

# LIST OF ABBREVIATIONS

AI: Artificial Intelligence

ASL: American Sign Language

BLEU: Bilingual Evaluation Understudy

CNN: Convolutional Neural Network

DFD: Data Flow Diagram

DHH: Deaf and Hard of Hearing

FNN: Feedforward Neural Network

ICT: Information and Communication Technology

IDE: Integrated Development Environment

ML: Machine Learning

MNIST: Modified National Institute of Standards and Technology

ROI: Region of Interest

SDLC: Software Development Life Cycle

WHO: World Health Organization

# LIST OF FIGURES

[Figure 1.1 Waterfall Model 3](#_Toc161012367)

[Figure 3.1: Use Case Diagram 6](#_Toc161012368)

[Figure 3.2 ER Diagram 9](#_Toc161012369)

[Figure 3.3: DFD LEVEL 0 10](#_Toc161012370)

[Figure 3.4 DFD LEVEL 1 11](#_Toc161012371)

[Figure 4.1: System Flow 12](#_Toc161012372)

[Figure 4.2: Sign-to-Speech Flow Diagram 13](#_Toc161012373)

[Figure 4.3: Speech-to-Sign Flow Diagram 14](#_Toc161012374)

[Figure 4.4: Analysis Module Flow Diagram 15](#_Toc161012375)

[Figure 4.6: High Level Design of Model 16](#_Toc161012376)

[Figure 4.7: Interface for Homepage 17](#_Toc161012377)

[Figure 4.8: Interface for Sign to Speech Page 18](#_Toc161012378)

[Figure 4.9: Interface for Speech to Sign Page 18](#_Toc161012379)

[Figure 4.10: Interface for Analysis Page 19](#_Toc161012380)

[Figure 4.11: Feedforward Neural Network 20](file:///C:\Users\97798\Desktop\signlink%20-%20updated11.docx#_Toc161012381)

[Figure 4.12: Convolutional Neural Network 22](#_Toc161012382)

[Figure 5.1: Test for detecting sign language using FNN 33](#_Toc161012383)

[Figure 5.2: Test for playing sign language according to the speech 34](#_Toc161012384)

[Figure 5.3: Loading the Home Page 35](#_Toc161012385)

[Figure 5.4: Loading of Sign-To-Speech Window using FNN 36](#_Toc161012386)

[Figure 5.5: Loading of Sign To Speech Window using CNN. 36](#_Toc161012387)

[Figure 5.6: Loading of Speech-To-Sign Window 37](#_Toc161012388)

[Figure 5.7: Loading of Analysis Window 37](#_Toc161012389)

[Figure 5.8: Confusion Matrix of FNN 39](#_Toc161012390)

[Figure 5.9: Training and Validation Loss of FNN Model 40](#_Toc161012391)

[Figure 5.10 Confusion Matrix of CNN 41](#_Toc161012392)

[Figure 5.11: Training and Validation Loss of CNN Model 41](#_Toc161012393)

[Figure 5.12: Comparison of Model Performance 42](#_Toc161012394)

[Figure 5.13: Comparison of Accuracy between Two Models 42](#_Toc161012395)

# LIST OF TABLES

[Table 3.1: Functional Requirements 7](#_Toc161012417)

[Table 3.2: Non-Functional Requirements 7](#_Toc161012418)

[Table 4.1: Overview of the Dataset 16](#_Toc161012419)

[Table 5.1: Tools Used 24](#_Toc161012420)

[Table 5.2: Test for Detecting Sign Language 32](#_Toc161012421)

[Table 5.3: Test for playing sign language according to the speech 34](#_Toc161012422)

[Table 5.4: Test for loading the application 35](#_Toc161012423)

# Introduction

## Introduction

At the current time according to WHO about 8% of the world's total population belongs to the DHH community and has a hard time communicating with speech users. Signalink is a desktop-based application that classifies and translates American Sign Language into text and speech and vice versa. It uses machine learning algorithms to classify the hand signs and then translates them by displaying the corresponding text and speech. For the classification, two different algorithms CNN and FNN were used.

The project also provides a dashboard where the two algorithms can be compared and visualized using the different charts. The batch size, epoch, and learning rate can be given by the user manually and see the comparison with the visual charts in real time.

## Problem Statement

Many people who use sign language have trouble communicating with people who don't understand sign language. This makes it hard for them to do everyday things like talk to others, get help, or access services. Sometimes they need a sign language interpreter, but that can be expensive and not always available. In addition, there are not many tools that can translate sign language quickly and accurately.

The purpose of the project is to recognize sign language and translate it into text and audio.

## Objective

The main objective of this project is:

• To create a system that translates sign language into text or spoken language and vice versa.

## Scope and Limitation

The scope of the project is to develop a system that can recognize the sign language and translate it. This system aims to translate the sign language into both text and voice and vice versa in real time used by the user.

The different limitation of our project are:

* With poor lightening condition it is hard to detect and recognize sign.
* Camera may not work properly in low light condition.

## Development Methodology

In this project waterfall method has been used as the development methodology. Each phase must be completed before into the next one in this software development lifecycle method and there is little to no overlap between the phases. The waterfall model in sign language translation project follows the following phases:

**Requirement gathering**

In this initial phase, the goal is to gather and document detailed requirements for the ASL translation system, such as the range of signs to be recognized, desired output formats (text or speech), performance expectations, and user interface preferences.

**System design**

Based on the gathered requirements, we will design the overall architecture and components of the ASL translation system. This phase will involve designing the Feedforward Neural Network model architecture, selecting appropriate computer vision techniques for hand tracking and feature extraction, and defining the translation and user interface components. Detailed design documents, including data flow diagrams, ER diagram will be produced.

**Implementation**

In this phase, the code is written for the entire system based on design specifications. The computer vision algorithms, the CNN and FNN model is implemented and the real time processing logic is developed for Sign language translation.

**Testing**

The unit testing is performed for individual components and modules is conducted. Similarly integration testing is performed to ensure that the system components work together as expected.

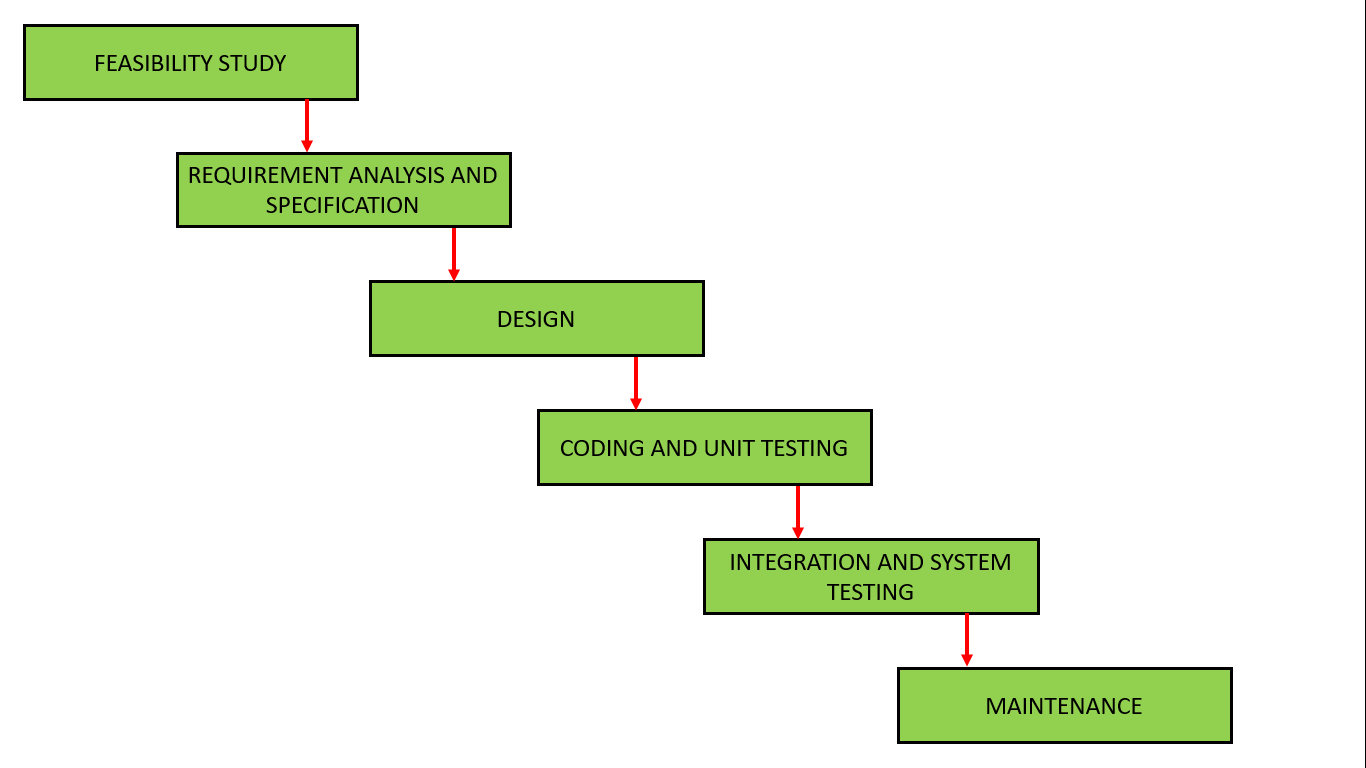


Figure 1.1 Waterfall Model

## Report Organization

We have organized out report in the following way:

Chapter 1: It includes the introduction section, the problem we attempted to solve, and the objectives and scope, development methodology for the project.

Chapter 2: It includes the background study of fundamental theories, general concepts and literature review of similar projects, theories and results by other researchers.

Chapter 3: It provides an overview of all the requirements along with system analysis and feasibility analysis of the system.

Chapter 4: It includes a detailed description of how the system was designed. It also includes the details of the algorithm used.

Chapter 5: It includes the tools we used to build the system and how the testing process was done.

Chapter 6: It includes the conclusion of the project and how we are further planning to make the system work sustainably.

# Background Study and Literature Review

## Background Study

Sign language is a visual-gestural language used by deaf and hard of hearing individuals for communication. It relies on hand shapes, facial expressions, and body movements to convey meaning. However, not everyone is proficient in sign language, leading to communication barriers and social isolation for deaf individuals.

According to WHO 8% of world’s total population belongs to DHH Community. Despite the technological advances there is still seems to be a barrier between the speech and sign user.

Recognition and classification of sign language using machine learning models involve several steps. Initially, data collection and proper labeling are essential. Subsequently, the data undergoes processing, and a suitable machine learning algorithm is selected. For our project, we have opted for two machine learning models: Feedforward Neural Network (FNN) and Convolutional Neural Network (CNN). Following the selection of the algorithm, the model is trained using the labeled data. Nodes, the fundamental units within a Feedforward Neural Network, receive input, undergo mathematical operations, and produce output. These nodes are organized into layers, including input, hidden, and output layers. Activation functions play a crucial role in determining the network's output by introducing non-linearities, facilitating the learning of complex patterns within the data. After training the model, performance evaluation is conducted using various evaluation metrics.

## Literature Review

Conducting a thorough literature review is essential to understanding the current landscape of machine learning application in sign language.

Machine learning enables the recognition and interpretation of sign language gestures, allowing for real-time translation into spoken language and vice versa. By analyzing video streams, machine learning algorithms can accurately interpret sign language gestures, enhancing accessibility. Additionally, it enables the development of gesture-to-speech interfaces, facilitating seamless communication. Through continuous learning and refinement, these systems can improve their accuracy and effectiveness over time. Ultimately, machine learning plays a pivotal role in breaking down communication barriers for deaf or hard-of-hearing individuals, promoting inclusivity and accessibility in society.

There have been several studies that have used CNN and FNN for image processing. There is a Study [1] where they developed a multi-layer fully connected neural network with a single hidden layer to recognize handwritten digits. Testing was carried out using the publicly accessible MNIST handwritten database, consisting of 28,000 digit images for training and 14,000 images for testing. Their artificial neural network achieved an impressive test accuracy of 99.60%.

Another [2] study they propose a novel deep learning approach aimed at detecting sign language, aiming to bridge this communication gap. Their methodology involves the creation of a dataset comprising 11 sign words, which they use to train a customized Convolutional Neural Network (CNN) model for real-time sign language detection. Preprocessing steps were applied to the dataset before training the CNN model. Their results demonstrate that the customized CNN model achieves impressive performance metrics, including 98.6% accuracy, 99% precision, 99% recall, and 99% f1-score on the test dataset.

The study [3] implements a Feedforward Neural Network (FNN) for image classification, aiming to enhance its structure by integrating the dropout method to prevent overfitting. The FNN is initialized with random uniform values and zero biases, incorporating ReLU and Softmax activation functions. MNIST-handwritten numbers dataset is used for evaluation Dropout is chosen as it efficiently prevents overfitting by randomly dropping neurons during training. The FNN with dropout achieves an average accuracy of 99.86% and a loss of 0.47%, outperforming the standard FFN's 98.13% accuracy and 9.15% loss. Comparatively, CNN achieves 99.26% accuracy and 2.39% loss. Dropout not only enhances accuracy but also reduces training time due to its iterative nature.

In conclusion, FNN and CNN have been successful in image classification tasks in several studies.

# Requirement Analysis and Feasibility Study

## System Analysis

### Requirement Analysis

For this project, the requirement analysis process aimed to identify the key features and functionality of the system, as well as any constraints or limitations that needed to be considered during development.

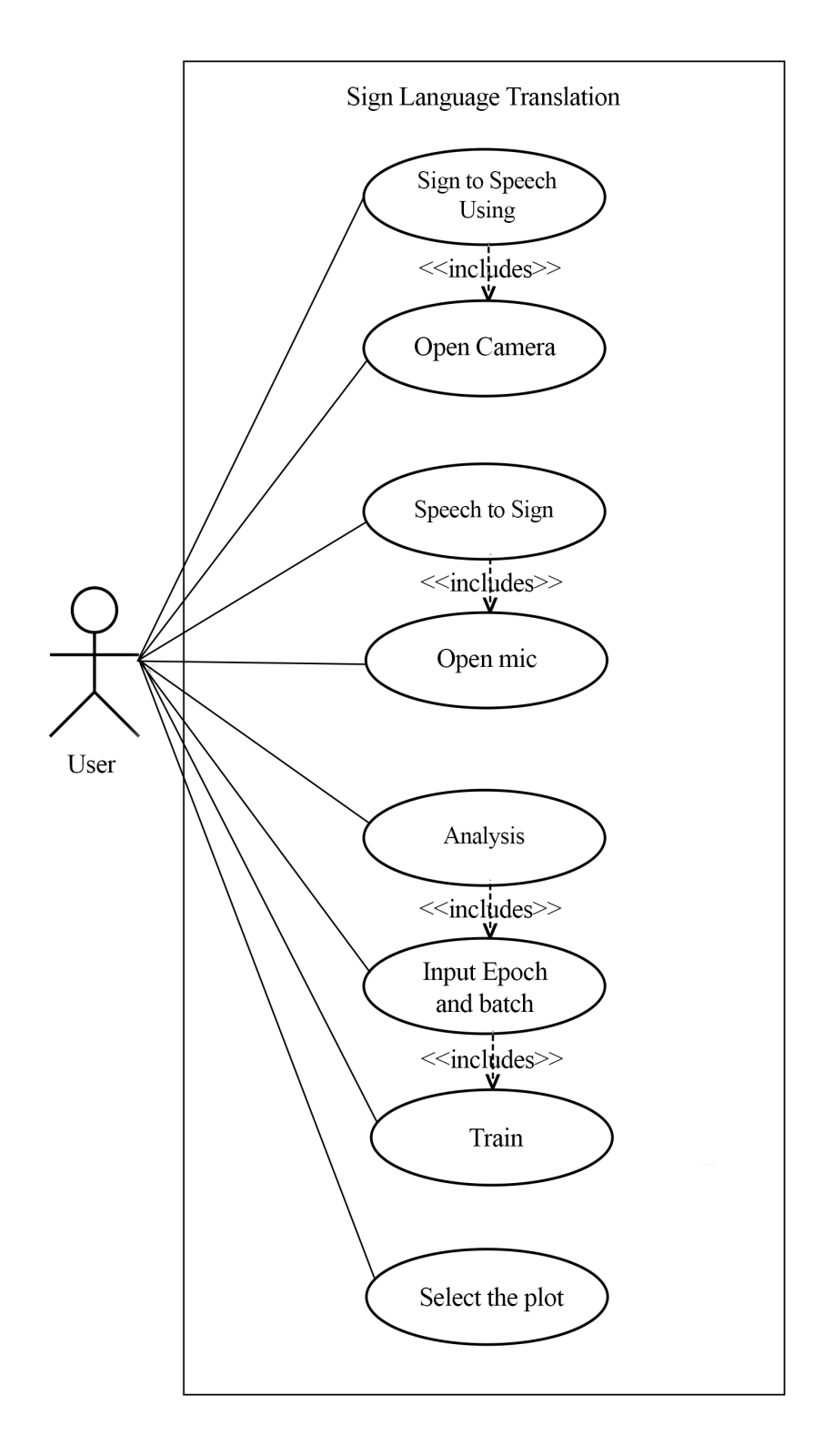


Figure 3.1: Use Case Diagram

**3.1.1.1 Functional Requirement**

Table 3.1: Functional Requirements

|  |  |  |
| --- | --- | --- |
| Req no. | Req. name | Req. Description |
| FR1 | Sign Language Recognition | The system should be able to recognize and interpret signs. |
| FR2 | Speech Synthesis | Convert sign language input into spoken language using speech synthesis |
| FR3 | Text Output | the system should be able to convert sign language input into written text. |
| FR4 | User Interface | Provide an intuitive and user-friendly interface for both input and output |
| FR5 | Real-time Processing | Ensure real-time processing of sign language input to provide immediate feedback |
| FR6 | Speech Recognition | The system should be able to recognize speech |
| FR7 | Speech to Sign Conversion | Translate recognized speech to sign language |

**3.1.1.2 Non-functional Requirement**

Table 3.2: Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| Req. No. | Req. name | Req. Description |
| NFR1 | Performance | The software should respond to user inputs and provide translations within a reasonable time frame, even under peak usage conditions. |
| NFR2 | Scalability | Application will go beyond a college project for  which necessary actions like upgrading server,  creating a team, will be taken care of. |
| NFR3 | Usability | Our system is simple to use which makes.  accessing desire feature a lot easier and faster.  Every UI component is arranged properly for  easy navigation and effective usage. |
| NFR4 | Maintainability | The application will be very simple to maintain because detailed documentation describing all of the system's components will be prepared.  This guarantees that future software developers and engineers will have no trouble ensuring the quality of our application. |

### Feasibility Study

**3.1.2.1 Technical Feasibility**

The technical feasibility of a sign language translator is boosted by the prevalence of cameras in today's mobile phones and computers, making it accessible to a broad audience. Analyzing sign language videos doesn't demand high-end computers, reducing costs. This makes developing and maintaining such a translator economically viable, as it utilizes existing hardware and minimizes the need for specialized equipment. Leveraging readily available technology allows for the implementation of sign language translation system sat a relatively low cost, enhancing their overall technical feasibility.

**3.1.2.2 Economic Feasibility**

The technical feasibility of a sign language translator is enhanced by the widespread availability of cameras in modern mobile phones and computers, making it accessible to a wide range of users. The processing requirements for analyzing sign language videos can be achieved with relatively modest computer systems, reducing the cost barrier. This makes the development and maintenance of a sign language translator economically feasible as it minimizes the need for specialized equipment.

**3.1.2.3 Operational Feasibility**

The sign language translator is operationally feasible as it utilizes data and algorithms to accurately translate sign language, making it practical and sustainable over time. The system is designed to be user-friendly and easily integrated into existing communication platforms, ensuring its ease of use and operational effectiveness.

**3.1.2.4 Schedule Feasibility**

A crucial aspect in the project's successful completion was teamwork. The project's objectives were attained in part because of the team members' efficient work distribution and smooth coordination. Because of this, the project was finished on time and according to a planned schedule, demonstrating that scheduling was feasible.

### Analysis

**3.1.3.1 Data Modeling**

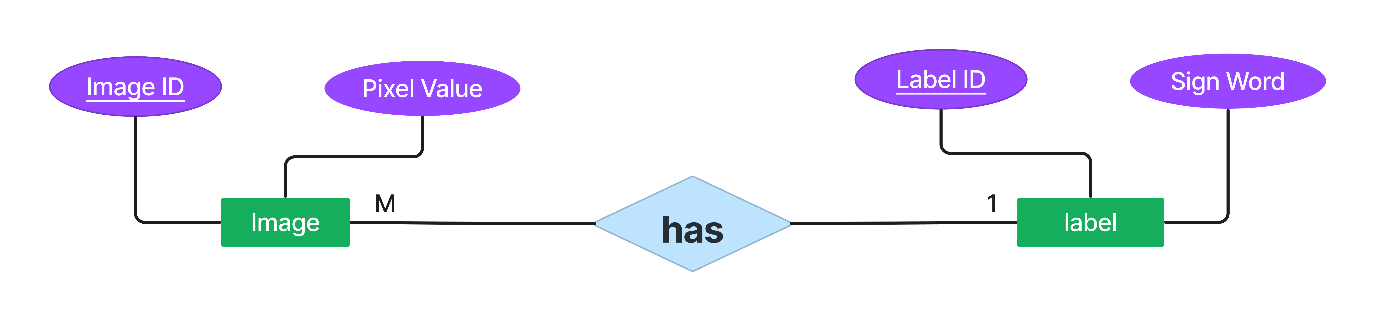


Figure 3.2 ER Diagram

The entity present are Image and label. Each image is considered as an entity. Each image entity would have attributes representing the pixel values of the image. In label entity the label have attribute i.e. label id and the word respective to the hand sign image. The relationship that is present is Has, here many image have one label.

**3.1.3.2 Process Modeling**

* **DFD Diagram**

DFD Level 0

The Level 0 DFD illustrates the user inputting hyperparameters (epoch, batch size, learning rate) into the system. The system processes hand signs and voice inputs, utilizing machine learning algorithms to generate sign language translations. It outputs predicted labels, voice translations, and corresponding videos, facilitating user interaction and comprehension. This high-level view outlines the flow of data, emphasizing the system's core functionalities and the exchange of information between users and the translation system.

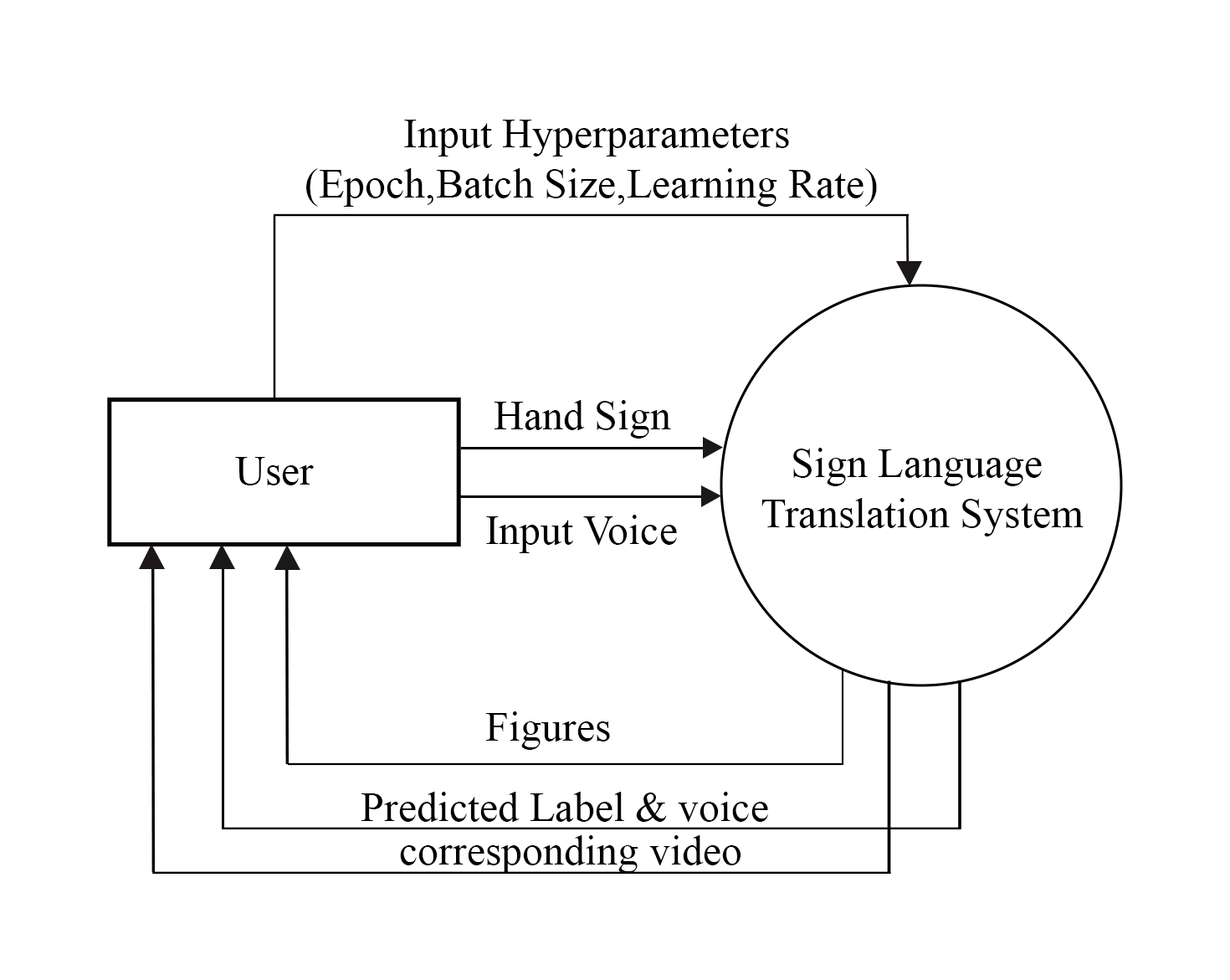


Figure 3.3: DFD LEVEL 0

DFD Level 1

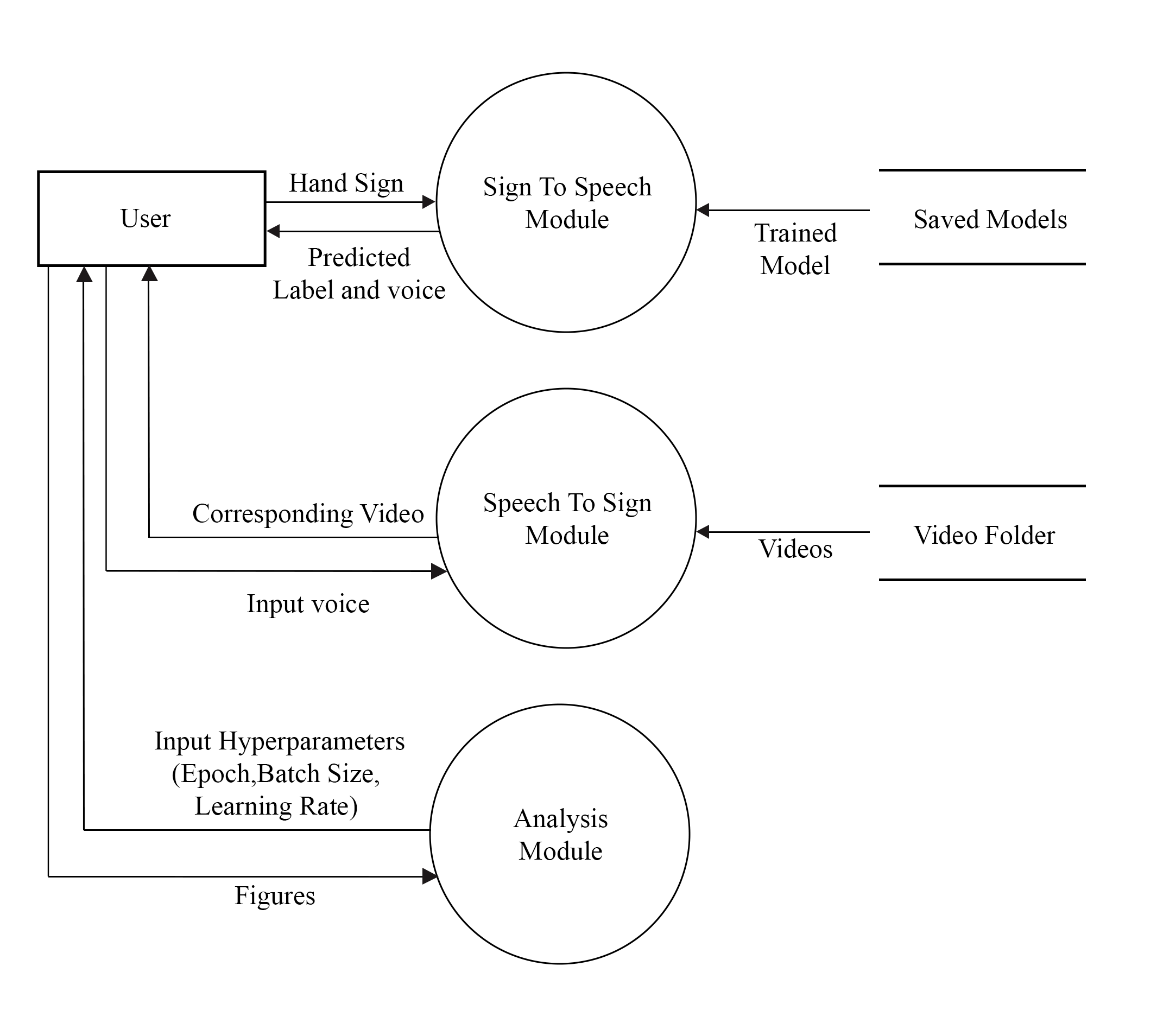
At Level 1 of the DFD, the Sign Language Translation System comprises three main modules: Sign to Speech, Speech to Sign, and Analysis, all initiated by user input of hyperparameters for optimal system configuration. The Sign to Speech module preprocesses hand sign data, extracting features and normalizing them before feeding them into dedicated machine learning models designed to translate sign language into speech. Simultaneously, the Speech to Sign module processes voice input by preprocessing it and converting it into textual form. This text data undergoes analysis and translation using specialized machine learning algorithms to generate corresponding sign language representations. The analysis model is used to compare between the FNN and CNN. Following the translation processes, the system integrates predicted speech outputs, sign language representations, and analysis results. These are then presented to users via the system interface, facilitating interaction and review

Figure 3.4 DFD LEVEL 1

# System Design

## Design

The application is layered on top of hand sign and detection and language detection mechanism, which checks for either hand sign to translate it into text and audio or checks for audio input to recognize language and translate it into hand sign.

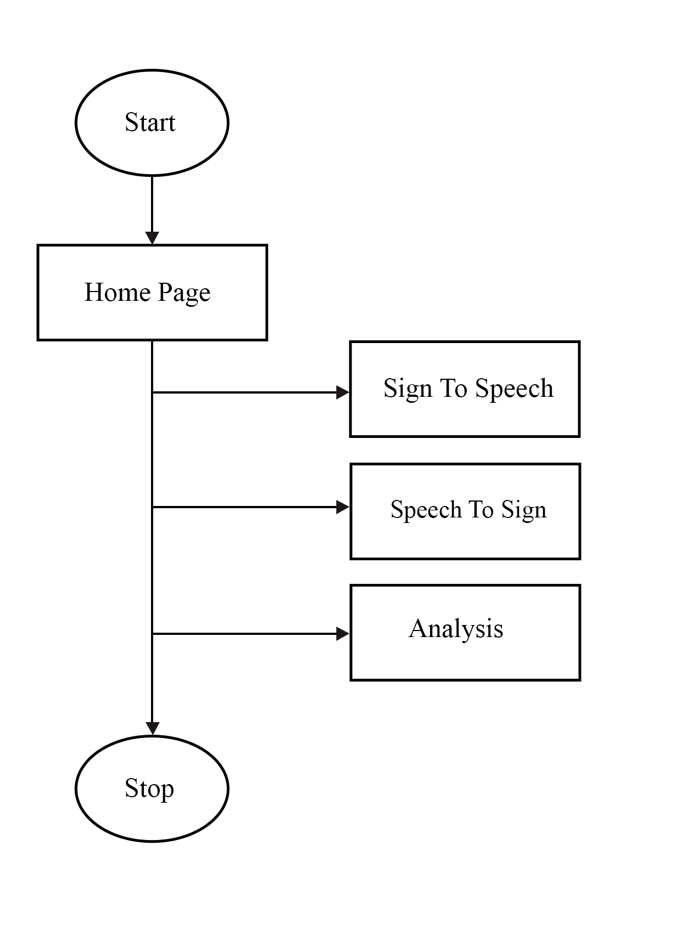


Figure 4.1: System Flow

At first the user is presented with visually appealing home page featuring three distinct options: Sign to Speech, Speech to Sign, and Analysis. This intuitive interface allows users to seamlessly navigate between functionalities for efficient communication and analysis within the ASL translation system.

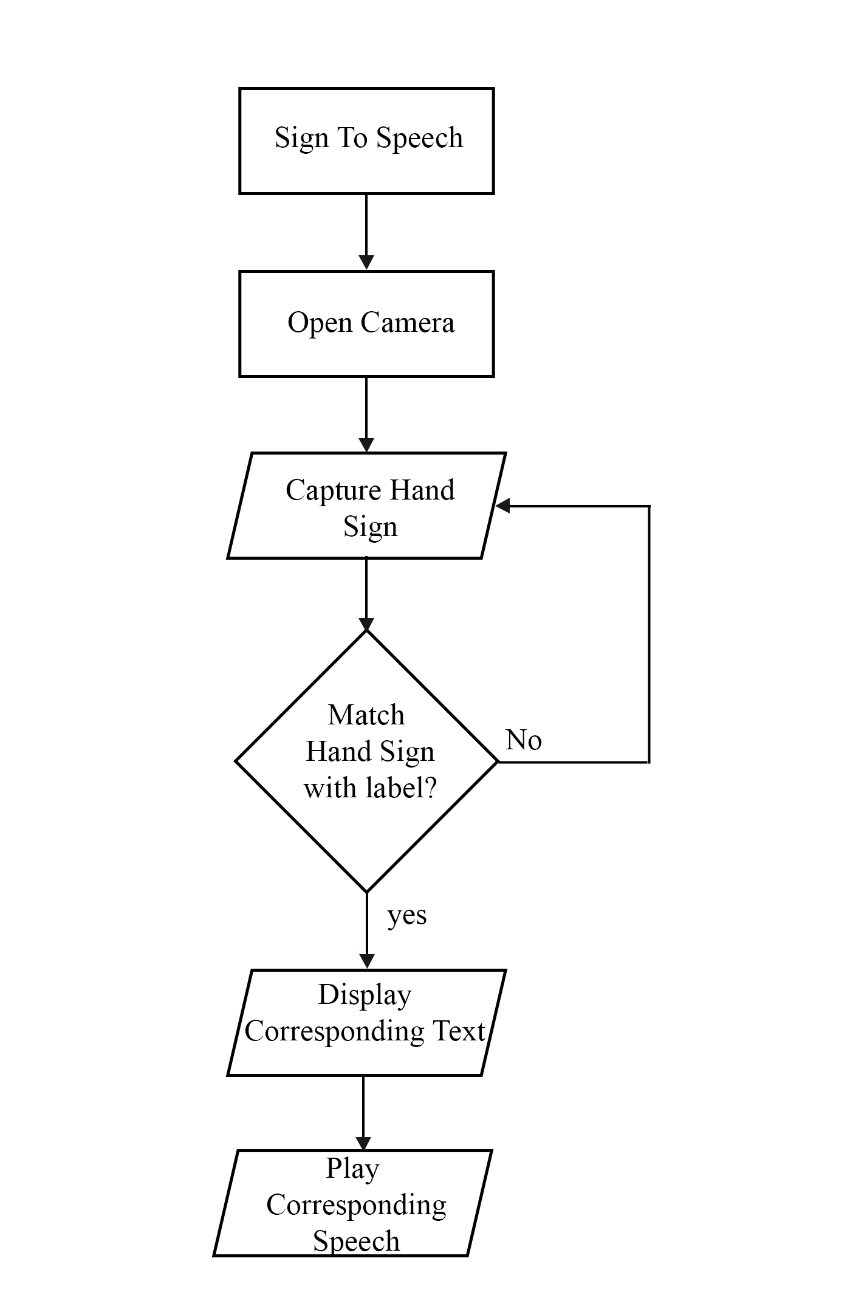


Figure 4.2: Sign-to-Speech Flow Diagram

In this workflow, the camera is initially opened to convert sign to speech. Once the camera is open, it captured the hand sign. If the hand sign matches with label to the corresponding text, it proceeds to the next step, which is to display the corresponding text. If the hand sign does not match, it goes back to capture hand sign step. After the corresponding text is displayed, the next step is to play the corresponding speech.

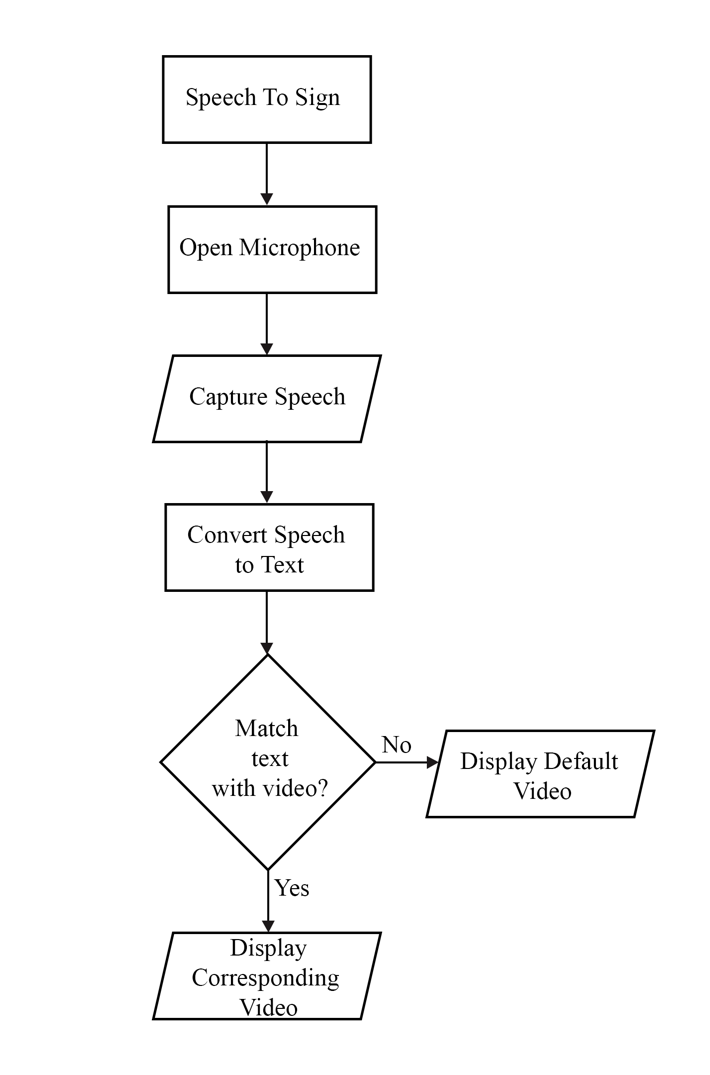


Figure 4.3: Speech-to-Sign Flow Diagram

In the Speech to Sign workflow, the first step is to open the microphone so that it can proceed to the next step, which is to capture speech. After the speech is captured, it is converted to text. Following the completion of the text conversion process, it matches the text with the video. If a match is found, it displays the corresponding video; otherwise, it displays the default video.

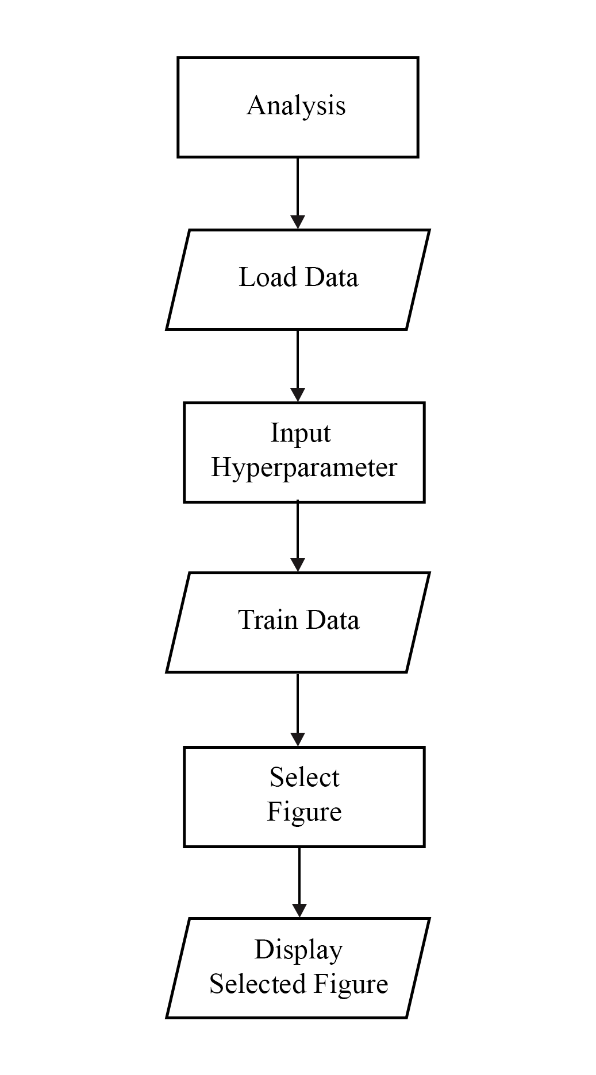


Figure 4.4: Analysis Module Flow Diagram

In the analysis workflow, the first step is to load the data in which we want to perform the analysis. After that we input the hyperparameter such as epoch, batch size and learning rate so that the data can be trained according to our need. Then we select the figure we like to display such as bar graph, confusion matrix, box plot and loss and accuracy plot.

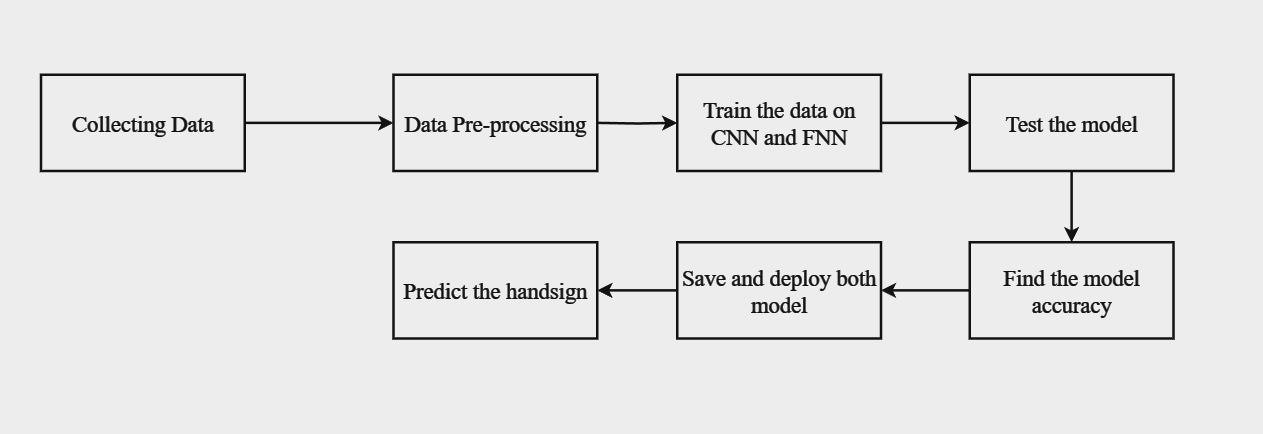
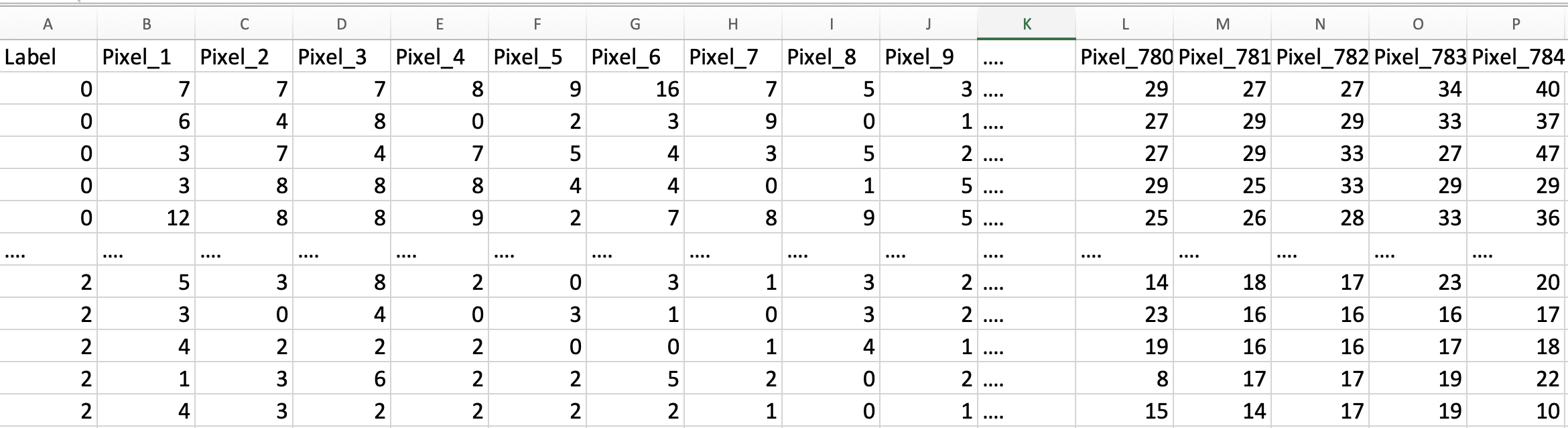


Figure 4.5: High Level Design of Model

* **Data Collection**

The dataset for our project is collected by ourselves. The data formatting for the project followed the structure of the MNIST dataset, which includes labels and image pixels. For training, data was collected for three labels, with 600 datasets available for each label. For testing, data for three labels was collected, with 200 datasets available for each label. Each row consists of one label and 784 pixels, as the image size is 28x28 pixels. A custom capture model is utilized to capture images, initially at a size of 200x200 pixels, which are subsequently resized to 28x28 pixels. These images are then converted to grayscale and transformed into CSV format. In the CSV file, each pixel is represented by a value between 0 and 255, where 0 denotes the darkest black and 255 represents the brightest white pixel.

Table 4.1: Overview of the Dataset



### Interface Design

Making an interface design before starting the front-end development is crucial. The project interface is made using Figma, the interface is designed to analyze the requirements of the project. The project consists of mainly 4 pages: The home page, Sign to Speech Page, Speech to Page, and Analysis Page.



Figure 4.6: Interface for Homepage

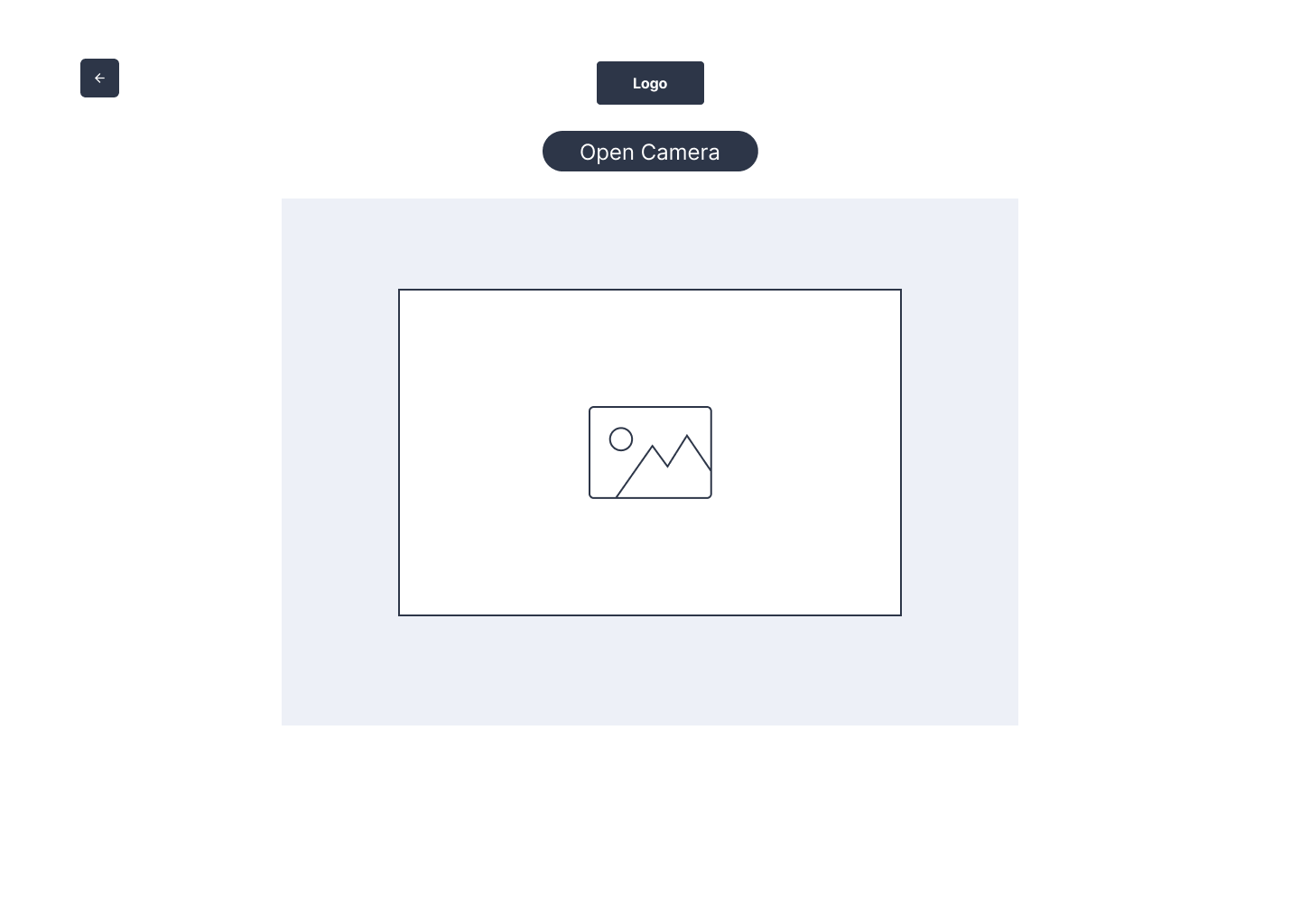


Figure 4.7: Interface for Sign to Speech Page

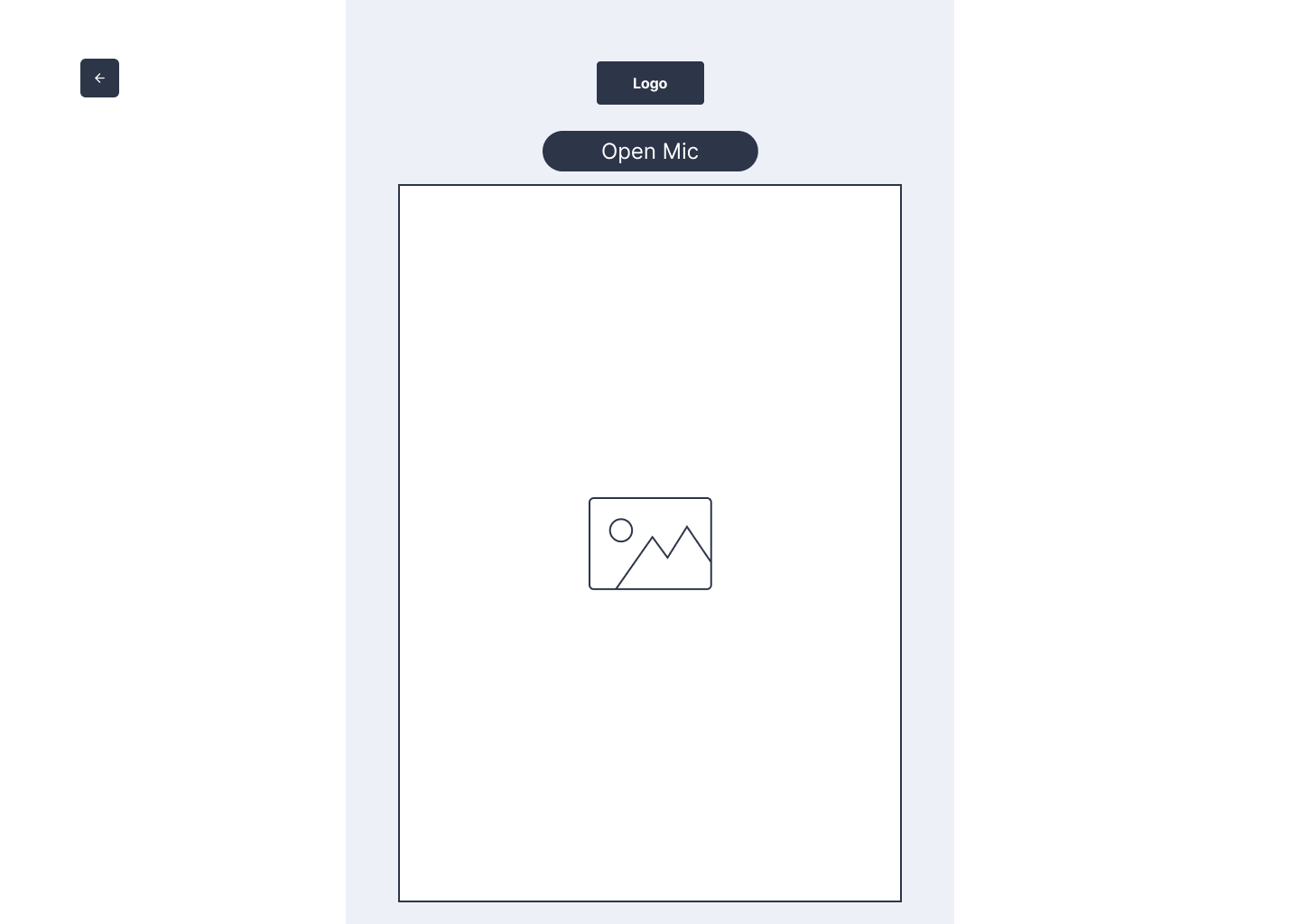


Figure 4.8: Interface for Speech to Sign Page

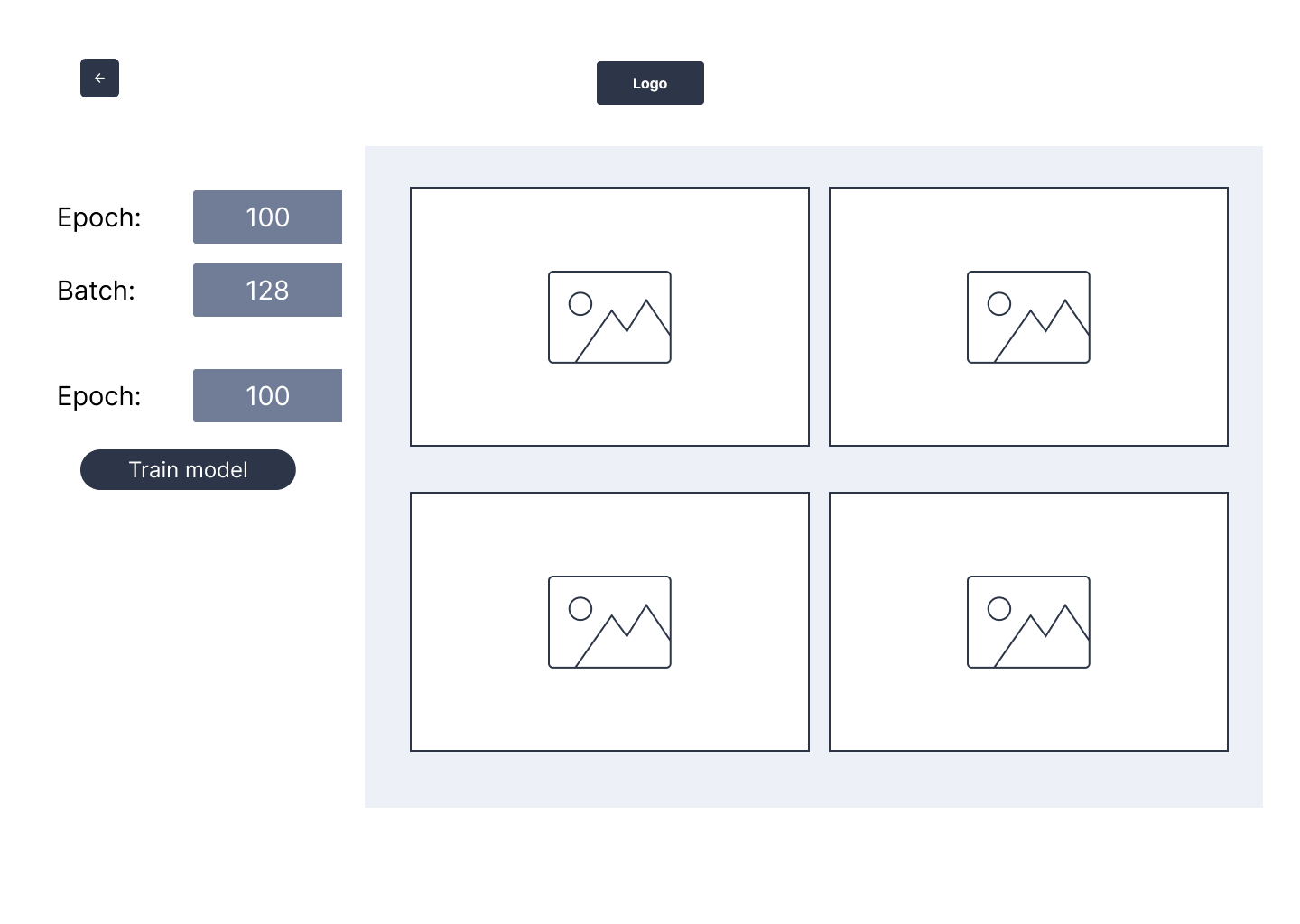


Figure 4.9: Interface for Analysis Page

## Algorithm Details

1. **Feedforward Neural Network**

The Feedforward Neural Network architecture consists of interconnected nodes organized into layers, including input, hidden, and output layers. Each node receives input signals, processes them through weighted connections, and applies an activation function to produce an output. During training, the network adjusts its weights based on the difference between predicted and actual outputs, minimizing the loss function through techniques such as gradient descent and backpropagation. In our ASL translation system, the FNN is trained on labeled sign language data to learn the mappings between input gestures and their corresponding meanings. [3]

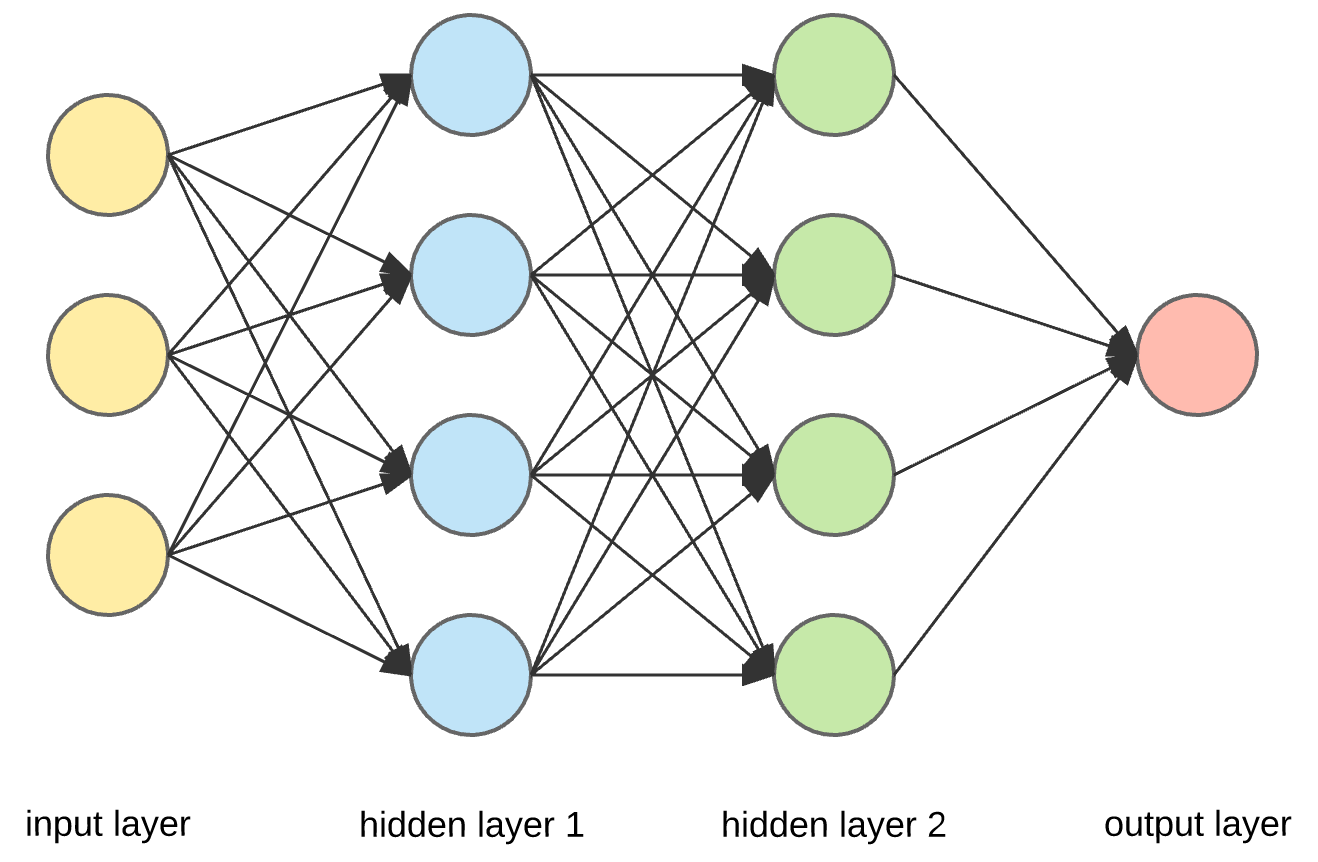


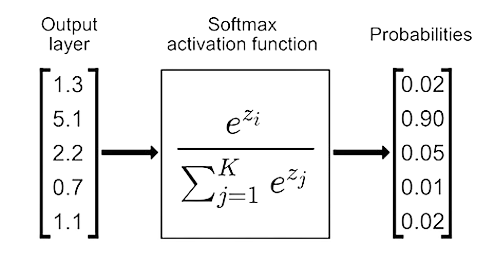
Figure 4.10: Feedforward Neural Network

The steps involved in FNN are as follow:-

**1. Data Preparation**: first we need a label dataset consisting of pixel value of each pixel of grayscale image ranging from 0 to 255

**2. Forward Pass**: The forward propagation in which the input layer is the number of pixel in our dataset. The second layer is the hidden layer which is obtained by applying weight and bias term to each pixel or neuron then activation function is applied so that it doesn’t become a linear function. The activation function used to add the complexity in this layer is ReLu function. The output layer is also obtained by adding weight and bias term along with weight and activation function. The activation function applied in the output layer is softmax activation function which gives probability for the label.

**Z[1] = W[1] X+ Z[1]  
A[1] = gReLU(Z[1])  
Z[2] = W[2] A[1]+ b[2]A[2] = gsoftmax(Z[2])**



**3. Backward Pass:** In back propagation we first get the prediction, find how much it deviated by actual label to give some sort of error. We also find how much each of the previous weight and bias term contributed to the error. The third part is updating the parameter. Then this part is propagated back and repeated again.

**dZ[2] = A[2] -Y  
dW[2] = dZ[2] A[1]TdB[2] = dZ[2]   
dZ[1] = W[2]TdZ[2].\*g[1]′ (z[1])  
dW[1] = dZ[2] A[0]T  
dB[1] = dZ[1]**

**4. Update parameters:** Updating parameters involves adjusting the weights and biases of the neurons to minimize the error between the predicted output and the actual target values during training. This process, known as backpropagation, aims to fine-tune the network's parameters to improve its ability to make accurate predictions or classifications.

**W[2] = W[2] – αdW[2]b[2] = b[2] – αdb[2]W[1] = W[1] – αdW[1]b[1] = b[1] – αdb[1]**

In the Feedforward Neural Network used in our system there is 784 neurons in input layer 10 in hidden layer and 3 in output layer.

**Convolutional Neural Network (CNN):**

Convolutional layers extract features from input images through convolution operations, while pooling layers reduce spatial dimensions, enhancing computational efficiency. Fully connected layers integrate the extracted features to make predictions. CNNs excel at capturing spatial hierarchies and patterns within images, making them ideal for hand gesture recognition in our project. [3]

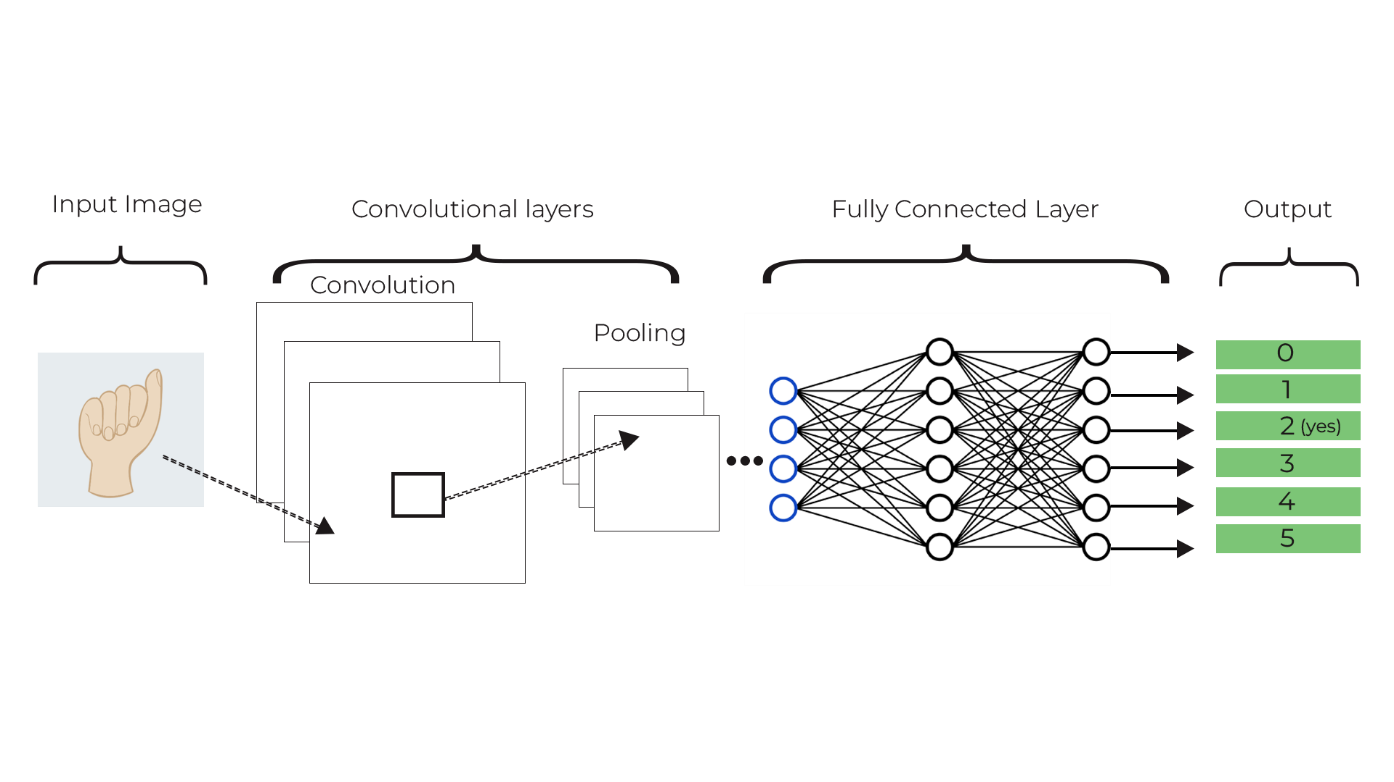


Figure 4.11: Convolutional Neural Network

The steps involved in CNN are as follow:-

**Input Image:** The input would be images of hand gestures representing different signs in the sign language. Each image is preprocessed than represented as a matrix of pixel values.

**Convolutional Layer:** The convolutional layer applies a set of filters(kernels) to the input data. These filters slide over the input image, performing element-wise multiplication and summation to produce feature maps. The filters act as feature detectors, identifying patterns and features at different spatial locations in the input images. As the network trains, the filters learn to detect low-level features such as edges, corners, shapes, curves etc. present in the MNIST dataset that we collected**.** MaxPooling was used to downsamples the feature maps obtained from the convolution layers and reduces the spatial dimensions. It takes the maximum value within each pooling window and make the representation more invariant to small translations and distortions in the input data from our csv file.

**(I∗K) (i,j)=∑m ∑ n I(m,n)⋅K(i−m, j−n)**

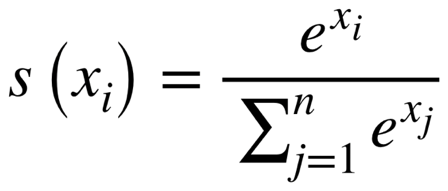
**I′ReLU (i,j)=max(0,I′ (i,j))**

**Y= max(W)**

**Fully Connected Layer:** After several convolutional and pooling layers, the high-level features are fed into fully connected layers. These layers consolidate the features learned by the convolutional layers and map them to the appropriate output classes. In the project, the fully connected layers help in recognizing complex patterns and relationships between different hand gestures.

**Output = activation (∑i x i ⋅w i +b)**

**Output:** The output layer represents the predictions made by the network. Each neuron in this layer corresponds to a sign language gesture. The network predicts the sign language gesture corresponding to the input image based on the activations of the neurons in the output layer.



# Implementation and Testing

## Implementation

For our project, we chose to use the Waterfall model because it offers a straightforward and systematic approach to software development. With this model, we can break down the project into clear and distinct phases, allowing us to focus on one aspect at a time. In this model, each stage of the software development life cycle must be finished before transitioning to the subsequent one, with minimal to no overlap between phases. For our Sign Language Translation , we chose to use the Waterfall model for our Sign Language system because it offers a straightforward and systematic approach to software development. With this model we can break down the project into clear and distinct phases, allowing us to focus on one aspect at a time. This helps to ensure that each phase, such as gathering requirements, designing the system, implementing features, testing for accuracy, deploying the final product.

### Tools Used

Different tools and technologies have been used to implement this application. They are listed in the table below:

Table 5.1: Tools Used

|  |  |  |
| --- | --- | --- |
| **Category** | **Tools** | **Description** |
| User Survey | Google Forms | versatile tool suitable for various purposes, such as conducting surveys, gathering feedback. |
| Diagram | Draw.io, Figma and Photoshop | This tools are utilized as a diagram tool to create visual representations of the system  architecture, UML diagram, and other technical diagrams. It aids in illustrating the  relationships and interactions between different components of the drowsiness detection  system, providing a visual guide for developers and stakeholders. |
| Language | Python | Python was used as the main programming language as we were already familiar with the language and it is popular for machine learning. |
| Code Editors | Jupyter Notebook, Visual Studio Code | Visual Studio Code was used as the main text editor and the code that required data visualization was done in Jupyter. |
| UI/UX | Figma | Figma was employed for making the wireframe and the main user interface of the system. |
| Documentation | Microsoft Office Package | MS Word was employed for documentation purposes. It provides a familiar desktop based environment for creating detailed project documentation, including specifications, user manuals, and other essential documentation. |
| 3D Model | DeepMotion, Blender | DeepMotion is a web application that tracks the video using AI and transfer it to 3d model . We used DeepMotion to convert our video into 3d model. The 3d model output was refined so we used Blender to refine the 3d model. |

### Modules Description

1. **Implementation of Feedforward Neural Network**

**Step 1:** Data is loaded, normalized and the parameters are initialized.

﻿def load\_data(file\_path):

   data = pd.read\_csv(file\_path)  
   data = np.array(data)

   np.random.shuffle(data)

   return data

def preprocess\_data(data):

   m, n = data.shape

   X = data[:, 1:].T / 255.0 # Normalize input data

   Y = data[:, 0]  
   return X, Y

def init\_params(input\_size, output\_size):

W1 = np.random.rand(10, input\_size) - 0.5  
   b1 =np.random.rand(10, 1) - 0.5  
  W2 =np.random.rand(output\_size, 10) - 0.5  
   b2 = np.random.rand(output\_size, 1) - 0.5

   return W1, b1, W2, b2

**# Load and preprocess data**

data = load\_data('./data/image28.csv')

X\_train, Y\_train=preprocess\_data(data)

**Step 2:** Then an activation function for the hidden layer that is ReLU function is applied and the activation function for the output layer which gives a probability is defined. The activation function helps to make the neurons non-linear.

def ReLU(Z):

return np.maximum(Z, 0)

def softmax(Z):

exp\_Z = np.exp(Z - np.max(Z)) # Subtracting max(Z) for numerical stability    return exp\_Z / np.sum(exp\_Z, axis=0)

**Step 3:** After defining the activation function forward pass is performed which used the activation function , multiples the activation function with the weight term and adds a bias term before passing to the next layer .

def forward\_prop(W1, b1, W2, b2, X):

Z1 = W1.dot(X) + b1  
 A1 = ReLU(Z1)  
 Z2 = W2.dot (A1) + b2  
 A2 = softmax(Z2)  
 return Z1, A1, Z2, A2

**Step 4:** After the forward pass backward pass is performed in which first we get the prediction and then compare with the label to find the error. We also find how much did the weight term and the bias term contributed towards those layer and go back more till the input layer.

﻿def backward\_prop(Z1, A1, Z2, A2, W1, W2, X, Y):

num\_classes = len(np.unique(Y))   
 m = X.shape[1]  
 one\_hot\_Y = one\_hot (Y, num\_classes)  
 dZ2 = A2 - one\_hot\_Y  
 dw2 = 1 / m \* dZ2.dot(A1.T)  
 db2 = 1 / m \* np.sum(dZ2, axis=1, keepdims=True)  
 dZ1 = W2.T.dot(dZ2) \* (Z1 > 0) # ReLU derivative  
 dw1 = 1 / m \* dZ1.dot(X.T)  
 db1 = 1 / m \* np.sum(dZ1, axis=1, keepdims=True)

return dw1, db1, dw2, db2

**Step 5:** Then when we find how much the bias and the weight term contribute towards the error we update the parameters accordingly.

def update\_params (W1, b1, W2, b2, dW1, db1, dW2, db2, alpha):

W1-=alpha \* dW1

b1-=alpha \* db1

W2-=alpha \* dW2

B2-=alpha \* db2

return W1, b1, W2, b2

**Step 6:** After that we define one\_hot function , prediction function and the accuracy function.

**Step 7:** Then a main function is defined which uses all the function and train the data and also compute the loss in each iteration or epoch.

﻿def one\_hot(Y, num\_classes):

one\_hot\_Y = np.zeros((num\_classes, Y.size))

one\_hot\_Y[Y, np.arange(Y.size)] = 1

return one\_hot\_Y

def get\_predictions (A2):  
 return np.argmax(A2, axis=0)

def get\_accuracy(predictions, Y):

return np.sum(predictions==Y) / Y.size

def gradient\_descent(X, Y, alpha, iterations):

W1, b1, W2, b2 = init\_params(X.shape[0], len(np.unique(Y)))

for i in range(iterations):

Z1, A1, Z2, A2 = forward\_prop(W1, b1, W2, b2, X)

dW1, db1, dW2, db2 = backward\_prop(Z1, A1, Z2, A2, W1, W2, X, Y)

W1, b1, W2, b2 = update\_params(W1, b1, W2, b2, dW1, db1, dW2, db2, alpha)

**# Calculate loss**

loss = compute\_loss(A2, Y)

if i % 10 == 0:

predictions = get\_predictions(A2)

accuracy = get\_accuracy(predictions, Y)

print(f"Iteration {i}: Loss = {loss:.4f}, Accuracy = {accuracy:.2f}")

return W1, b1, W2, b2

**Step 8:** After that prediction is made and tested if the prediction was correct or not by comparing the label we get with the actual label.

W1, b1, W2, b2 = gradient\_descent(X\_train, Y\_train, alpha=0.1, iterations=500)

﻿def test\_prediction(index, X, Y, W1, b1, W2, b2):

current\_image = X[:, index, None]  
 prediction = make\_predictions(X[:, index, None], W1, b1, W2, b2)

label = Y[index]  
 print("Prediction:", prediction)  
 print("Label:", label)

current\_image = current\_image.reshape((200, 200))\*255

plt.gray()  
plt.imshow(current\_image, interpolation='nearest')  
plt.show()

def make\_predictions(X, W1, b1, W2, b2):

\_› \_, \_, A2 = forward\_prop(W1, b1, w2, b2, x)

predictions = get\_predictions (A2)  
 return predictions

**Step 9:** The code saves model parameters (W1, b1, W2, b2) as 'mode6.pkl'. Subsequently, it evaluates predictions four times using test\_prediction. For each evaluation, the function utilizes training data (X\_train, Y\_train) and the saved model parameters.

**# Save the model**  
save\_model(W1, b1, W2, b2, 'mode6.pkl')

**# Test predictions**  
test\_prediction(0, X\_train, Y\_train, W1, b1, W2, b2)  
test\_prediction(1, X\_train, Y\_train, W1, b1, W2, b2)

test\_prediction(2, X\_train, Y\_train, W1, b1, W2, b2)  
test\_prediction(3, X\_train, Y\_train, W1, b1, W2, b2)

1. **Implementation of CNN**

**Step 1:** The train and test data is loaded from the CSV file.

**# Load training and testing data**

train = pd.read\_csv('./data/image28.csv')

test = pd.read\_csv('./data/image28Test.csv')

**Step 2:** Then the dataset is spiltted into training and testing set. It allocates 30% of the data for testing while keeping 70% for training.

**# Split data into training and testing sets**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(images, labels, test\_size=0.3, random\_state=101)

**Step 3:** Then the neural network model is constructed. It includes convolutional layers with ReLU activation, followed by max-pooling layers for feature extraction.

model = Sequential()

model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.20))

model.add(Dense(num\_classes, activation='softmax'))

**Step 4:** The model is compiled after the model construction process.

**# Compile the model**

model.compile(loss='categorical\_crossentropy',optimizer=Adam(),metrics=['accuracy'])

**Step 5:** In this step the model is trained using the training data (x\_train and y\_train). It specifies parameters such as batch size, number of epochs, and verbosity level for monitoring progress.

**# Train the model**

history = model.fit(x\_train, y\_train,

batch\_size=batch\_size,

epochs=epochs,

verbose=1,

validation\_data=(x\_test, y\_test))

**Step 6:** Then the model is evaluated and the prediction is made on the image.

**# Evaluate the model on test data**

test\_labels = test['Label']

test.drop('Label', axis=1, inplace=True)

test\_images = test.values

test\_images = np.array([np.reshape(i, (28, 28)) for i in test\_images])

test\_images = np.array([i.flatten() for i in test\_images])

test\_labels = label\_binrizer.fit\_transform(test\_labels)

test\_images = test\_images.reshape(test\_images.shape[0], 28, 28, 1)

**# Make predictions**

y\_pred = model.predict(test\_images)

## Testing

Testing involves assessing and confirming the functionality of the developed sign language application to ensure it accurately interprets and translates sign language gestures and movements. It aims to determine if the actual sign language recognition and translation outputs align with the expected results for various sign language inputs.

### Unit Testing

In this project, we have performed the unit testing in different module by checking the unit performance in different condition. We test each part to ensure it works correctly, just like checking that each sign is accurate and clear. we make sure the whole system understands sign language accurately.

Table 5.2: Test for Detecting Sign Language

|  |  |
| --- | --- |
| Objective | Detect hand and recognize corresponding sign “Yes” |
| Action | User showed “yes” sign |
| Expected Result | “yes” in text and audio |
| Actual Result | The application produced an output “yes” in both text and audio |
| Conclusion | The test was successful |

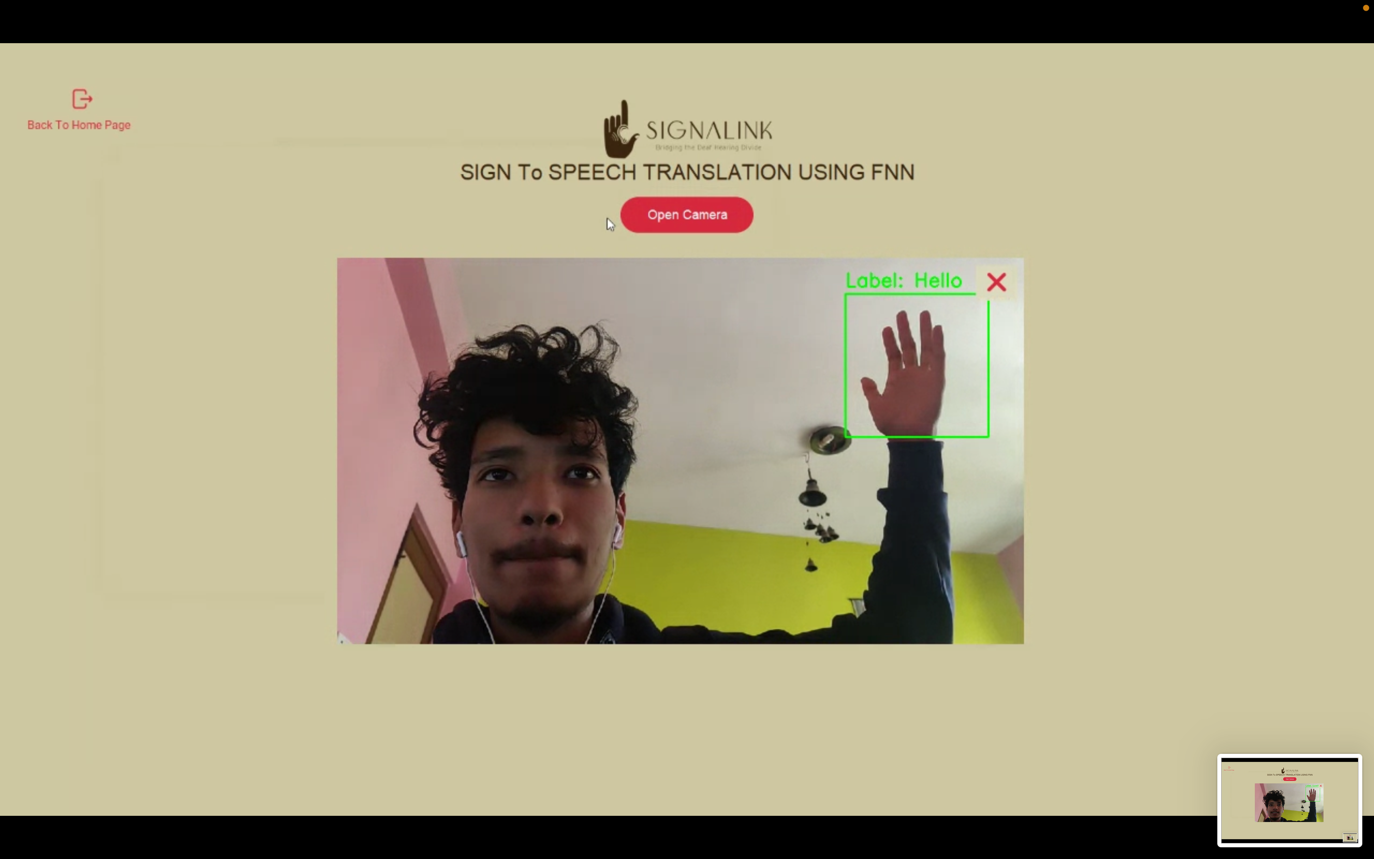


Figure 5.1: Test for detecting sign language using FNN

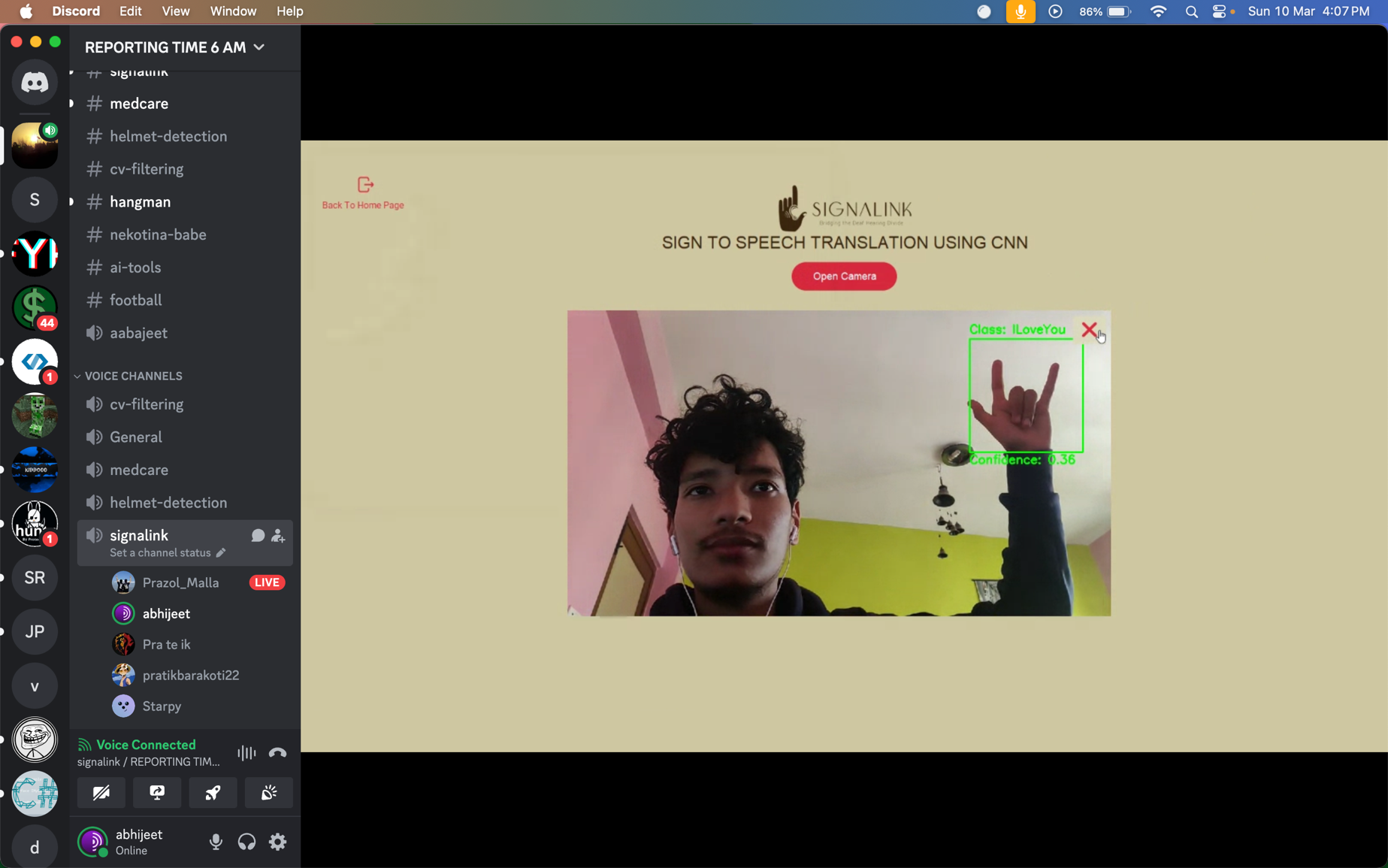


Figure: Test for detecting sign language using CNN

Table 5.3: Test for playing sign language according to the speech

|  |  |
| --- | --- |
| Objective | Detect the speech and showing corresponding sign language |
| Action | User said “Hello” |
| Expected Result | “Hello” sign in 3D Model |
| Actual Result | The application produced an output “Hello” sign in 3D Model |
| Conclusion | The test was successful |

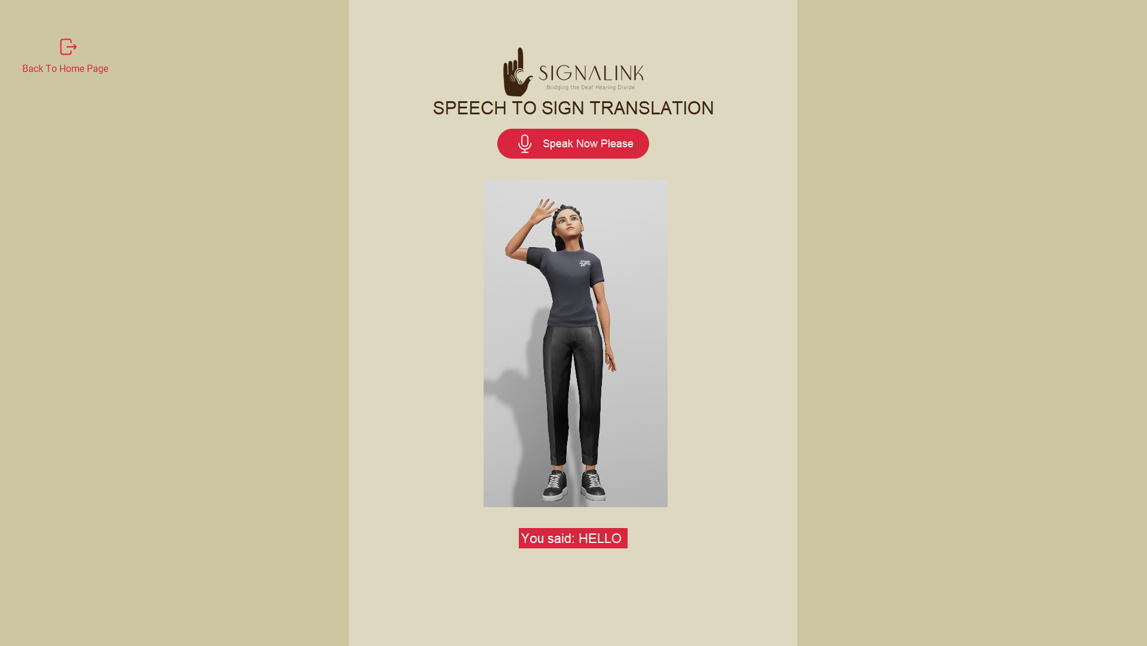


Figure 5.2: Test for playing sign language according to the speech

### System Testing

System testing in a sign language recognition project involves evaluating the entire system as a whole to ensure it meets its specified requirements and functions correctly in its intended environment. In sign language recognition system, testing involves capturing sign language gestures through input devices like cameras, processing the data, recognizing the signs accurately, and providing appropriate output or responses.

Table 5.4: Test for loading the application

|  |  |
| --- | --- |
| Objective | Opening the application |
| Action | The application was run through terminal command. |
| Expected Result | The application should load properly. |
| Actual Result | The application loaded properly. |
| Conclusion | The test was successful |



Figure 5.3: Loading the Home Page

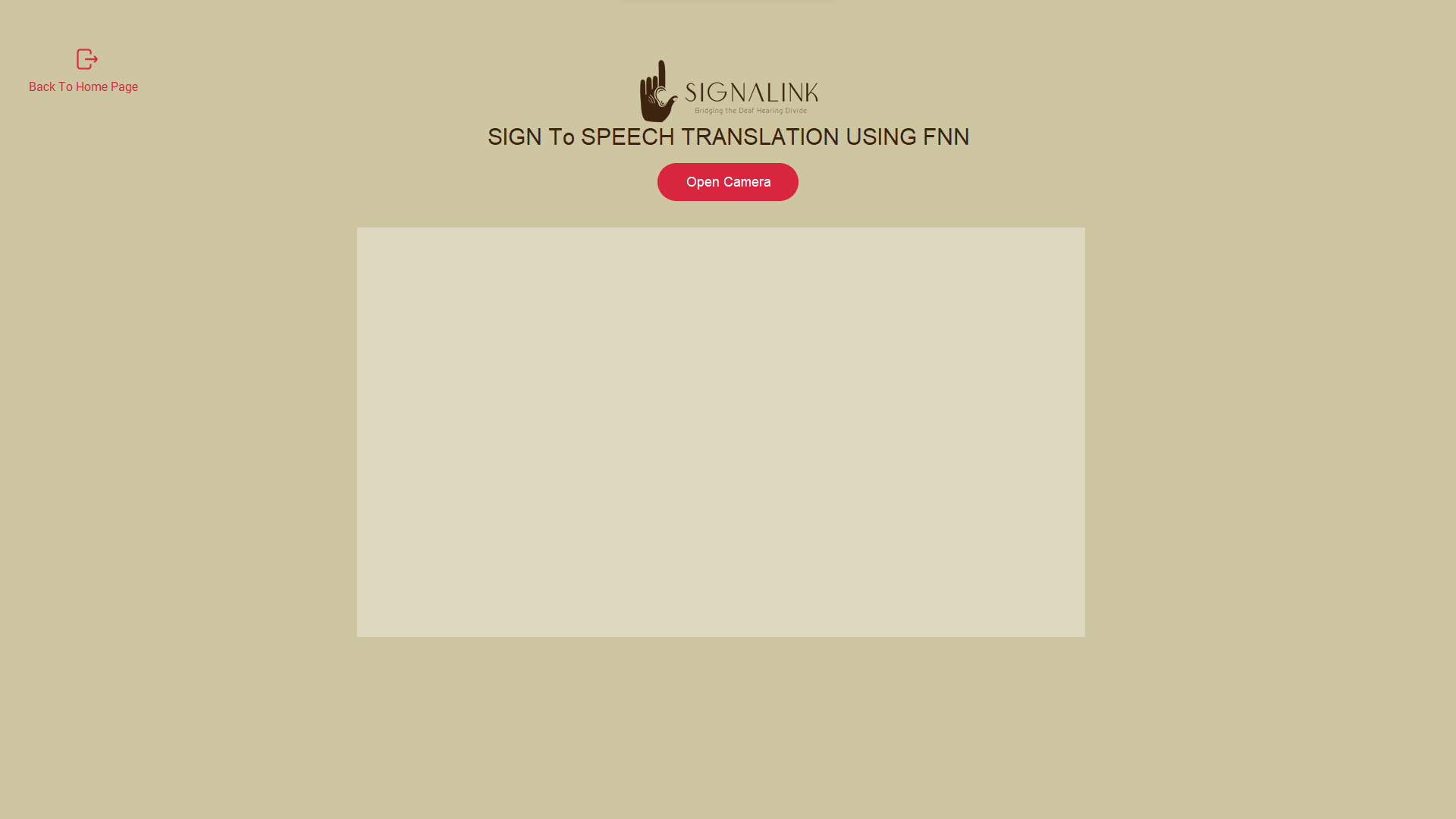


Figure 5.4: Loading of Sign-To-Speech Window using FNN

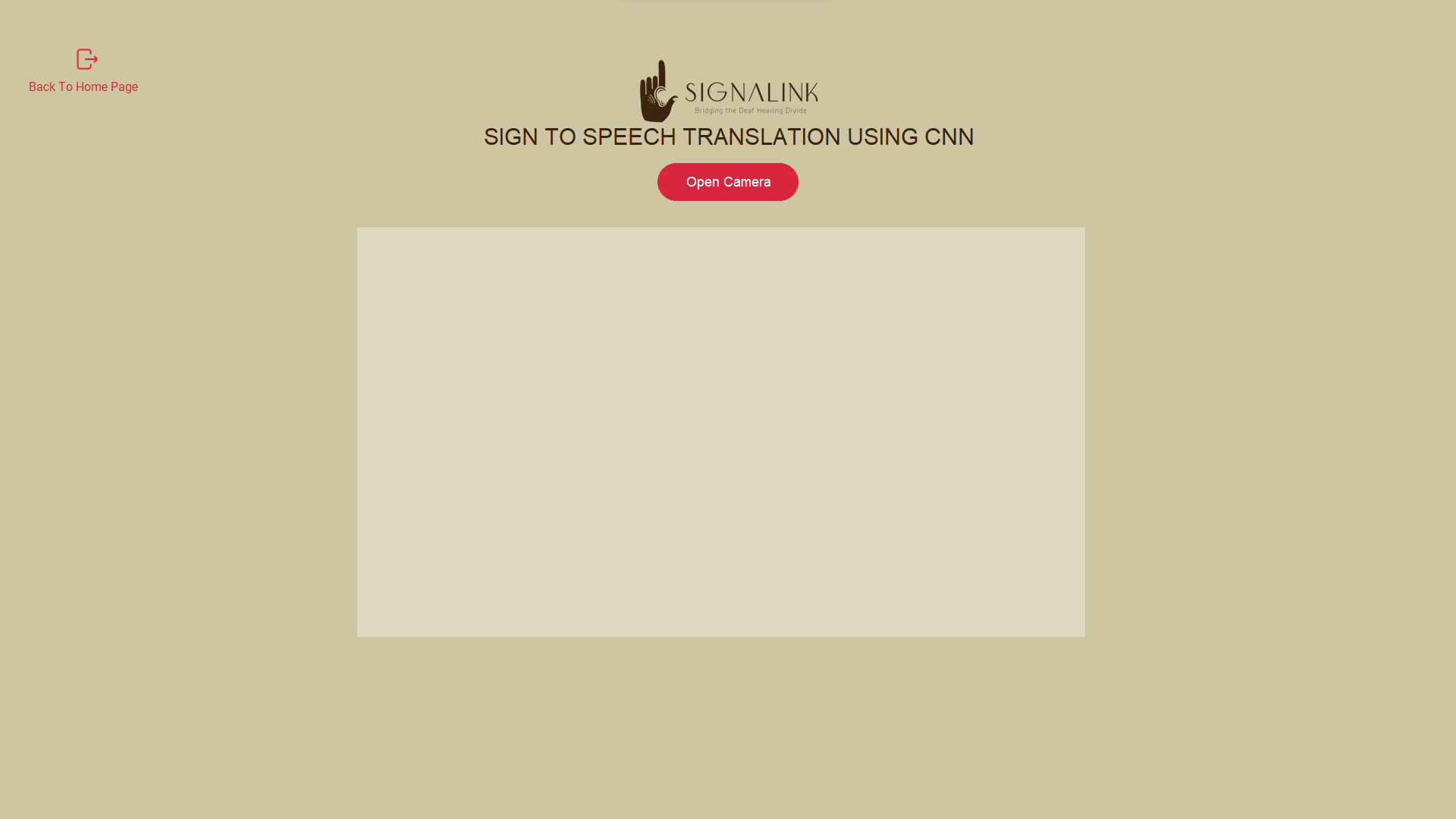


Figure 5.5: Loading of Sign To Speech Window using CNN.

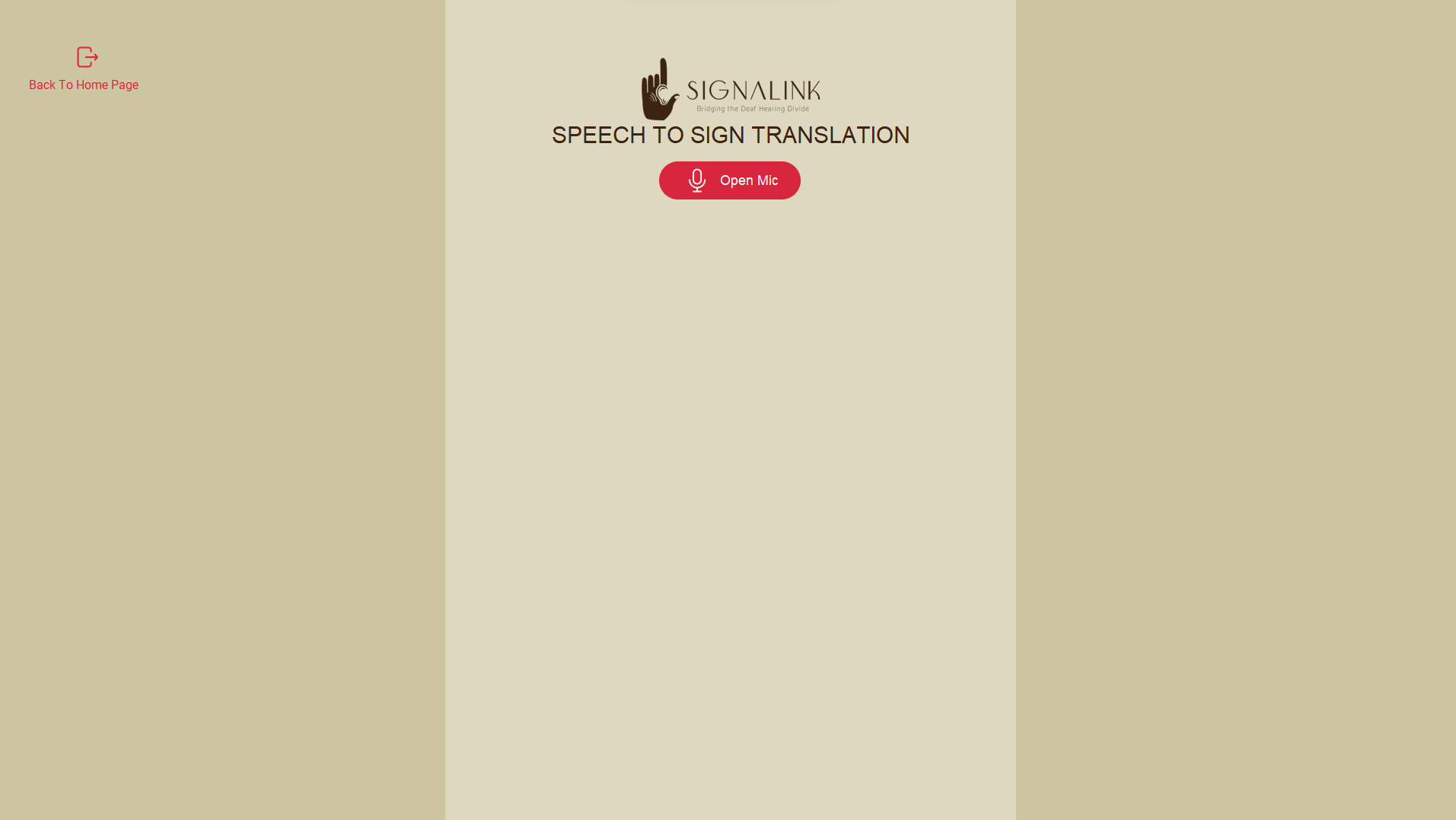


Figure 5.6: Loading of Speech-To-Sign Window



Figure 5.7: Loading of Analysis Window

## Result Analysis

The system was tested through unit testing and proved to be effective in executing its intended functions. The results showed that the project was able to meet its goals, but there is still room for improvement in terms of expanding the system's capabilities and increasing community involvement.

### Evaluating Accuracy

In machine learning, accuracy is a common metric used to evaluate the performance of a classifier model. Accuracy measures the proportion of correctly classified instances among all instances in the dataset. To calculate accuracy, the first step is to divide the dataset into two parts: a training set and a test set. The training set is used to train the model, while the test set is used to evaluate the model's performance.

In classifier model the most common measure to evaluate accuracy are:

• Precision: The precision measures the proportion of correctly identified sign language gestures or movements among all the gestures or movements that the model predicted as belonging to a particular sign.

The formula for precision is:

Precision = True Positives / (True Positives + False Positives)

For Label 1 in FNN from figure 5.8,

Precision = 252 / (252+29+74) = 70%

Recall: The recall measures the proportion of correctly identified sign language gestures or movements among all the actual sign language gestures or movements present in the dataset.

The formula for recall is:

Recall = True Positives / (True Positives + False Negatives)

For Label 1 in FNN from figure 5.8,

Recall = 252 / (252+37+11) = 84%

• F1 score: The F1 score provides a single metric that balances the trade-off between precision (accurately identifying sign language gestures) and recall (detecting most of the actual sign language gestures present). The formula for F1 score is:

F1 score = 2 \* (Precision \* Recall) / (Precision + Recall)

**Evaluating Accuracy for Feedforward Neural Network**

The best hyperparameters for the Feedforward Neural Network model were found to be an epoch of 10 with a learning rate of 0.01. The accuracy was 76% for the training dataset and 81% for the testing dataset. Sometimes, the confusion matrix misclassified label 1 and label 2, while label 0 was heavily misclassified as label 1 and label 2.

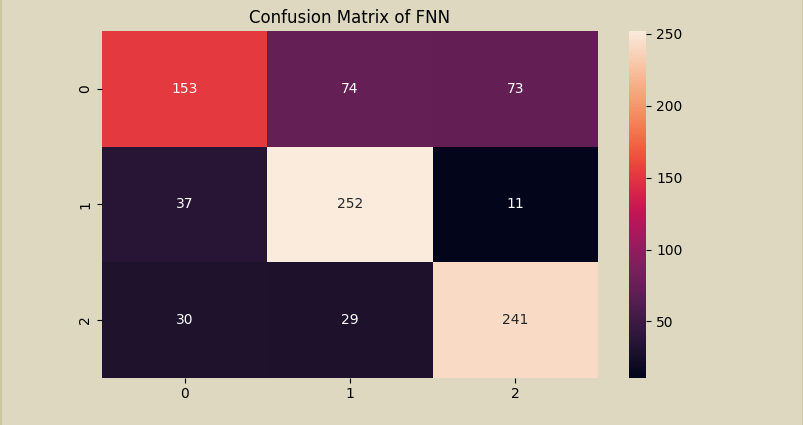


Figure 5.8: Confusion Matrix of FNN

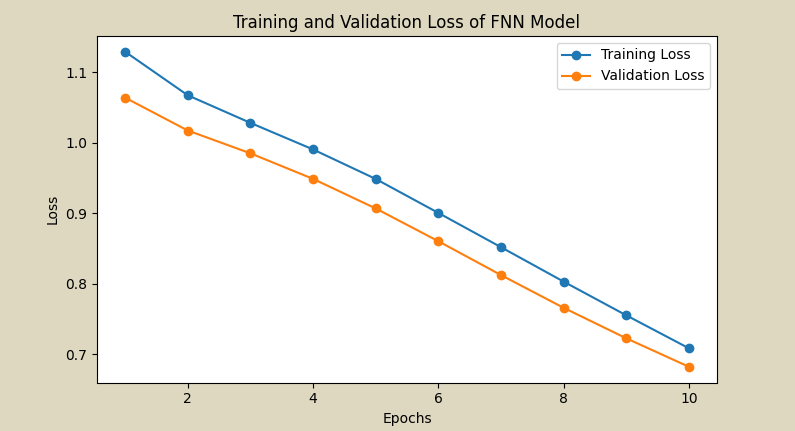


Figure 5.9: Training and Validation Loss of FNN Model

**Evaluating Accuracy for Convolutional Neural Network**

The best hyperparameters for the Feedforward Neural Network model were found to be an epoch of 20 with a learning rate of 0.01. The accuracy was 74% for the training dataset and 79% for the testing dataset. Label 1 and 2 was heavily misclassified as label 0

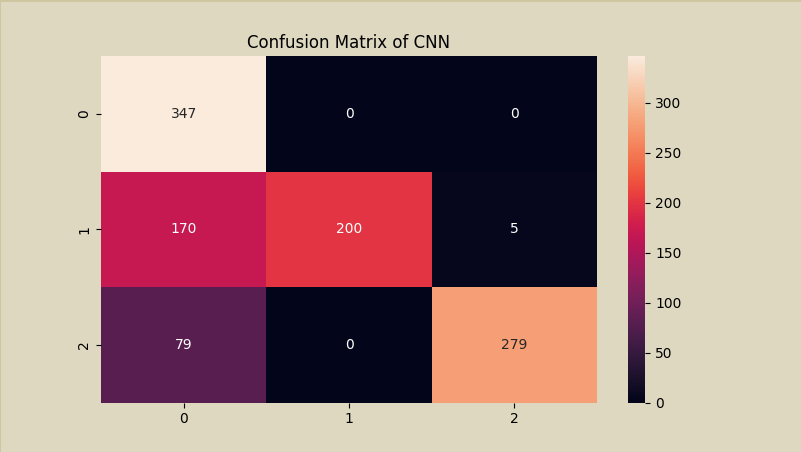
file:///C:/Users/97798/Documents/ViberDownloads/0-02-03-0d388554507407faabc651bd6d3a606c501d30153de2932e330b02a91ee812f4\_825326099fcf300.mp4

Figure 5.10 Confusion Matrix of CNN

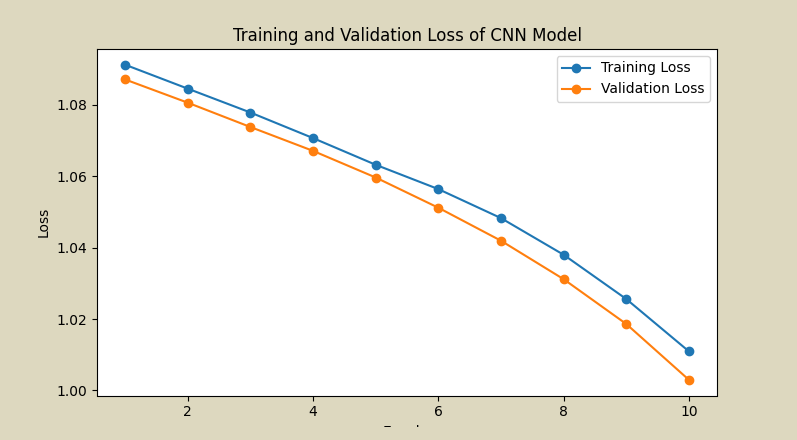


Figure 5.11: Training and Validation Loss of CNN Model

**Comparison of Model**

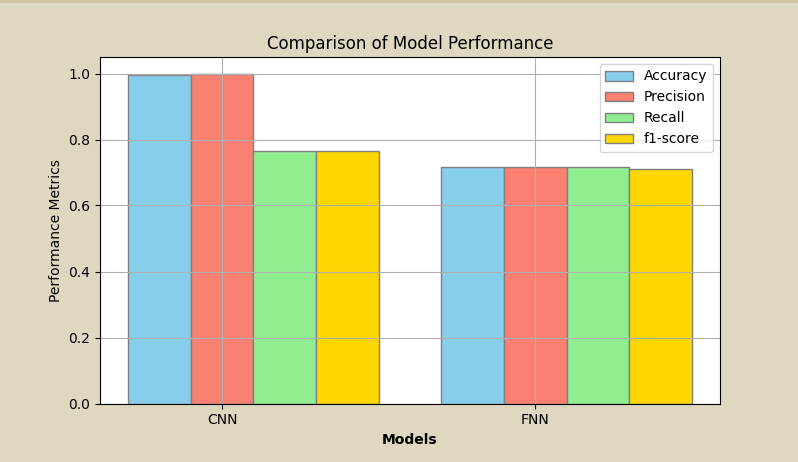


Figure 5.12: Comparison of Model Performance

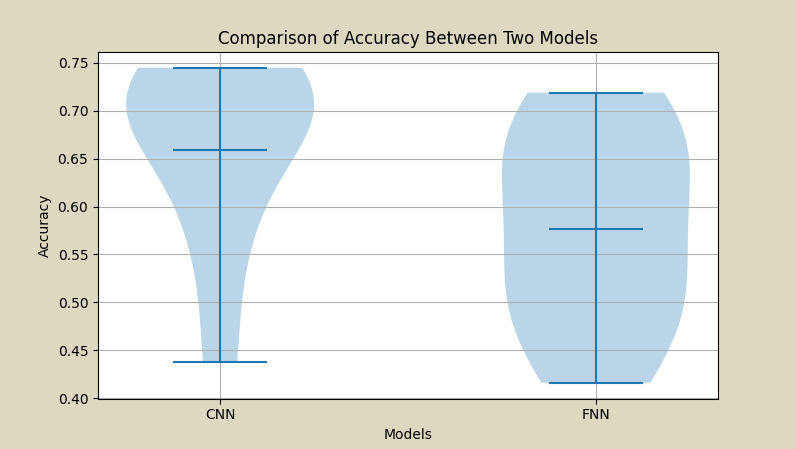


Figure 5.13: Comparison of Accuracy between Two Models

# Conclusion and Future Improvements

## Conclusion

In conclusion, the project classifies and translates the hand sign into text and speech using a Feedforward Neural Network and convolutional neural network algorithm. The result shows that the model effectively classifies and translates the sign language. The CNN algorithm proves to be a more effective tool for classifying the hand sign because of the convolution layer and is also robust to overfitting. Moreover, the model was trained on the custom data that we collected which mimics the MNIST dataset.

The project is also able to visualize the accuracy, precision, recall, and f1 score through different charts like bar graphs, line graphs, etc. The project is also able to show the comprehension and the visualization of the data through the user input epochs, batch size, etc. Overall, the project classifies and translates sign language to text and speech and also speech to sign.

## Future improvement

There are several enhancements that can be applied to this system. Those achievable within the budget and time limitations include:

**a. Enhanced Data Collection:** Improved methods for collecting and analyzing data can enhance the system's accuracy, enabling it to recognize a broader range of hand signs.

**b**. **Enhanced User Interfaces:** Upgrades to user interfaces can simplify system navigation, making it more user-friendly and enabling easier access to necessary information.

**c. Phrase Recognition:** The system can be developed to recognize phrases or longer sentences, enhancing communication and making it even more seamless and straightforward.

# References

|  |  |
| --- | --- |
| [1] | K. T. Islam, G. Mujtaba, R. G. Raj and H. F. Nweke, "Handwritten digits recognition with artificial neural network," *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T),* pp. 1-4, 2017. |
| [2] | M. Saiful, A. Isam, H. Moon, R. Tammana, M. Das, M. Alam and A. Rahman, "Real-Time Sign Language Detection Using CNN," 2022. |
| [3] | D. Basel, "CONVOLUTIONAL NEURAL NETWORK-BASED SIGN LANGUAGE TRANSLATION SYSTEM," vol. 9, pp. 47-57, 7 June 2020. |
| [4] | C. Lee, K. K. Ng, C.-H. Chen, H. Lau, S. Chung and T. Tsoi, "American Sign Language Recognition and Training Method with Recurrent Neural Network," *Expert Systems with Applications,* vol. 167, p. 114403, 04 2021. |