PROJECT REPORT

SBS CS 01 04 30 C 00168

"Sign.AI: Sign Language Recognition and Translation"

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATION

YEAR (2021-2023)

Submitted By

Satya Sangram Pattnaik	Kartik Kumar	Narendra Tiwari	Deepak
Roll No:210544	Roll No: 210529	Roll No: 210534	Roll No: 210520

Submitted to

Dr. Keshav Singh Rawat

Associate Professor

Central University of Haryana

Department of Computer Science & Information Technology

Student's Declaration

We, hereby declare that the project entitled "**Sign.AI: Sign Language Recognition and Translation**" submitted in the partial fulfillment of requirement for award of the degree of **Master of Computer Application** to **Central University of Haryana**, is our authentic work carried out during the 4th semester of our course under the supervision of **Dr. Keshav Singh Rawat**, Associate Professor, Department of Computer Science & Information Technology. The project has not formed the basis for award of any other degree, associate ship, fellowship or any other similar title.

Satya Sangram Pattnaik	Kartik Kumar	Narendra Tiwari	Deepak
Roll No: 210544	Roll No: 210529	Roll No: 210534	Roll No: 210520
Session: 2021-2023	Session: 2021-2023	Session: 2021-2023	Session: 2021-2023

Central University of Haryana

Department of Computer Science & Information Technology

This is to certify that this report entitled "**Sign.AI: Sign Language Recognition and Translation**" is a bonafide record of the project presented by **Satya Sangram Pattnaik** (210544), **Kartik Kumar** (210529), **Narendra Tiwari** (210534), **Deepak** (210520) Session 2021-2023 are students of Master of Computer Application at Department of Computer Science and Information Technology, Central University of Haryana. They have completed the project work under my supervision.

Date: …/…/…

Dr. Keshav Singh Rawat
Associate Professor,
Head of the Department,
Computer Science and Information Technology,
Central University of Haryana, Mahendragarh

ACKNOWLEDGEMENT

First and foremost, We would like to express our sincere gratitude to **Dr. Keshav Singh Rawat,** Associate Professor and Head of the Department of Computer Science and Information Technology, Central University of Haryana, for his unwavering support, knowledge, and encouragement during the course of our project. His opinions and suggestions were really helpful in forming this project and raising its caliber.

In addition, We want to thank the Central Department personnel and faculty for providing the tools, facilities, and academic atmosphere that made this project go smoothly. Their encouragement and dedication to creating a supportive learning atmosphere have been crucial to our success.

In closing, we would like to extend our sincere appreciation to everyone who contributed—small or big—to the success of this initiative. Your assistance, direction, and encouragement have been invaluable, and we are really appreciative of the chance to collaborate with such extraordinary people.

Thank you.

Satya Sangram Pattnaik	Kartik Kumar	Narendra Tiwari	Deepak
Roll No: 210544	Roll No: 210529	Roll No: 210534	Roll No: 210520

PREFACE

Sign languages (also known as signed languages) are languages that use the visual-manual modality to convey meaning, instead of spoken words. It is a complete language with its own grammar and vocabulary. Sign languages are not universal and are usually not mutually intelligible, although there are also similarities among different sign languages.

The technique of translating sign language motions into text or voice is known as sign language recognition. By facilitating their ability to interact with hearing people, this technology has the potential to enhance the lives of deaf people.

This project aims to develop a sign language recognition system for **Indian Sign Language** (**ISL**). The system will be able to recognize a variety of ISL gestures and generate text messages and also translate it into other languages. In this deep learning project, hand sign data that was manually created taking reference from the ISL website and was fed into a deep neural network that was trained to classify hand signs and potentially deduce their meaning. The system was evaluated on a variety of metrics, including accuracy, precision, recall and has simple graphical user interface that makes it user friendly.

INDEX

Chapter 1

Introduction

1.1 About Project
1.2 Aims and Objectives

1. INTRODUCTION

1.1. About Project

There are an estimated 14 million deaf people in India, and ISL is their primary means of communication. However, ISL is not widely understood by hearing people, which can make it difficult for deaf people to participate fully in society. Indian Sign Language (ISL) is a sign language used by hearing and speech impaired people in India. It is a visual-gestural language that uses hand shapes, facial expressions, and body language to convey meaning.

The Indian Sign Language Recognition project aims to develop a system that can automatically recognize ISL signs. This system would be a valuable tool for deaf people, as it would allow them to communicate with hearing people more easily. It would also be a valuable tool for teachers and interpreters, as it would allow them to learn and teach ISL more effectively. The project will use a variety of techniques, including computer vision, deep learning, and natural language processing. The system will be trained on a large dataset of ISL signs. The project is still in its early stages, but the team has made significant progress. The system has been able to recognize a number of ISL signs with high accuracy. The next step is to continue training the system on a larger dataset of ISL signs, and to evaluate it on a more challenging test set.

The team is confident that the system has the potential to make a significant impact on the lives of deaf people in India, and it could help to break down the barriers that prevent deaf people from participating fully in society.

1.2. Aims and Objectives

- Recognition of Indian Sign Language.
- Conversion of Sign Language into text.
- Translation into various languages.

Chapter 2

System Analysis

2.1 Model Requirement Specifications

- 2.1.1 General Description
- 2.1.2 System Objectives
- 2.1.3 User Requirements
- 2.1.4 Software Specifications
- 2.1.5 Hardware Specifications

2.2 Software Tools

2. SYSTEM ANALYSIS

2.1. Model Requirement Specifications

The requirement specification includes all the software, hardware, functional and non-functional requirements of the system.

2.1.1. General Description

Sign.AI is a Sign Language Recognition and Translation application built on top of MediaPipe Framework and it uses Computer Vision to Detect Hand Signs from a users video capture device. Sign.AI is accurate and also has the capability to autocorrect words and phrase sentences using GingerIT. Sign.AI also provides Hindi Translation for predictions with the help of argostranslate.

2.1.2. System Objectives

The project's main goal is to recognize sign language and translating it into text in various languages so that hearing-impaired people can freely interact with the bulk of the population who do not understand sign language.

2.1.3. User Requirements

This section covers all the functional and non-functional requirements that makes the app more user friendly and easy to use.

- **Accuracy –** Recognizing the signs accurately.
- **Speed –** Faster speed enables the user to keep pace with the other person.
- **Light –** Lighter system which can be easily installed.
- **User friendly –** The user should not face any difficulty while running the system.
- **Less resource requirement –** It should run using minimum resource.
- **Informative –** How to use instructions must be provided.
- **Robust –** Should adapt to any unnatural conditions.
- **Free to use –** User should get it free of charge or at minimal cost.

2.1.4. Software Specifications

- Windows 10/11
- Python 3.8.16
- Tensorflow 2.10.0
- Mediapipe 0.10.0
- OpenCV 4.7.0.68
- Tkinter

2.1.5. Hardware Specifications

- **CPU: >=**Intel i3
- **RAM: >=**8GB
- **Video Capturing Device: >=**5 mp, >=30fps
- **Storage: >**80 MB (available)
- **GPU** (optional)

2.2. Software Tools

- **Visual Studio Code**

  Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS.

- **Jupyter Notebook**

  The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

- **Pycharm**

  PyCharm is an integrated development environment used for programming in Python. It provides code analysis, a graphical debugger,

Chapter 3

System Design

3.1 Data Collection

3.2 Designing

- 3.2.1 Design Overview
- 3.2.2 System Representation
- 3.2.3 Key Technologies

3.3 File Structure

3. SYSTEM DESIGN

3.1. Data Collection

Because the techniques used in traditional machine learning and deep learning projects were so sophisticated, it took a long time for the engineers to construct the model. Because they were forced to code every implementation by hand due to the lack of current libraries, a significant chunk of the development process was focused on creating the ideal model to fit the data. There are several cutting-edge libraries available nowadays that implement complicated functionality and make them accessible to users. This shortens the amount of code that must be written and makes implementation easier. Because of this, gathering and preparing the project's data is the most crucial activity in contemporary data science projects.

Preprocessed datasets are available on many online platforms, such as Kaggle, making it much simpler for data engineers to work with the data. Sometimes you have to get information independently from numerous organisations or websites. Web scraping is a technique for gathering organised data from the web by crawling through the webpage's html code.

Because there was no available dataset that met our requirements for this particular project, we choose to generate the data ourselves. Learning the movements required in the Indian sign language system was made much easier by the website for Indian Sign Language. On the basis of it, we manually took images to construct our own dataset. OpenCV library was used to fetch pictures from webcam and stored it in numpy array. OpenCV captures frames from a video stream and stores it as a numpy array.

Mediapipe library for hand recognition is used to detect hand in that frame. Mediapipe is a library developed by google for detecting certain objects. Mediapipe hand recognition model can detect hands in a frame it detects 21 keypoints on each hand and there are certain function that return the x, y coordinates of all those 21 points. These points are stored in keypoint.csv file along with the sequence number that represent the gesture for that particular sign. 1000-2000 image were collected for each symbol for accurate training of the model. Corresponding gesture names are taken as user input and stored in a separate label.csv file along with the gesture id.

The hand landmark model bundle detects the keypoint localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds.

The hand landmarker model bundle contains a palm recognition model and a hand landmarks recognition model. The Palm recognition model locates hands within the input image, and the hand landmarks recognition model identifies specific hand landmarks on the cropped hand image defined by the palm recognition model.

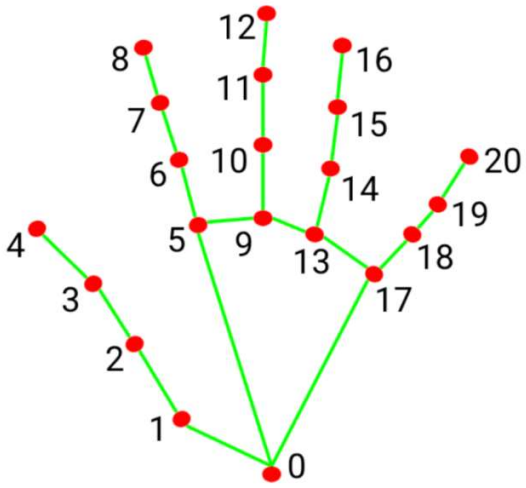

Fig 3.1. Hand Knuckle Points

0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fig 3.2. KeyPoint labels

Following are the images of signs of all letters and alphabets of Indian sign language system, The model has been trained on multiple instances of these images.

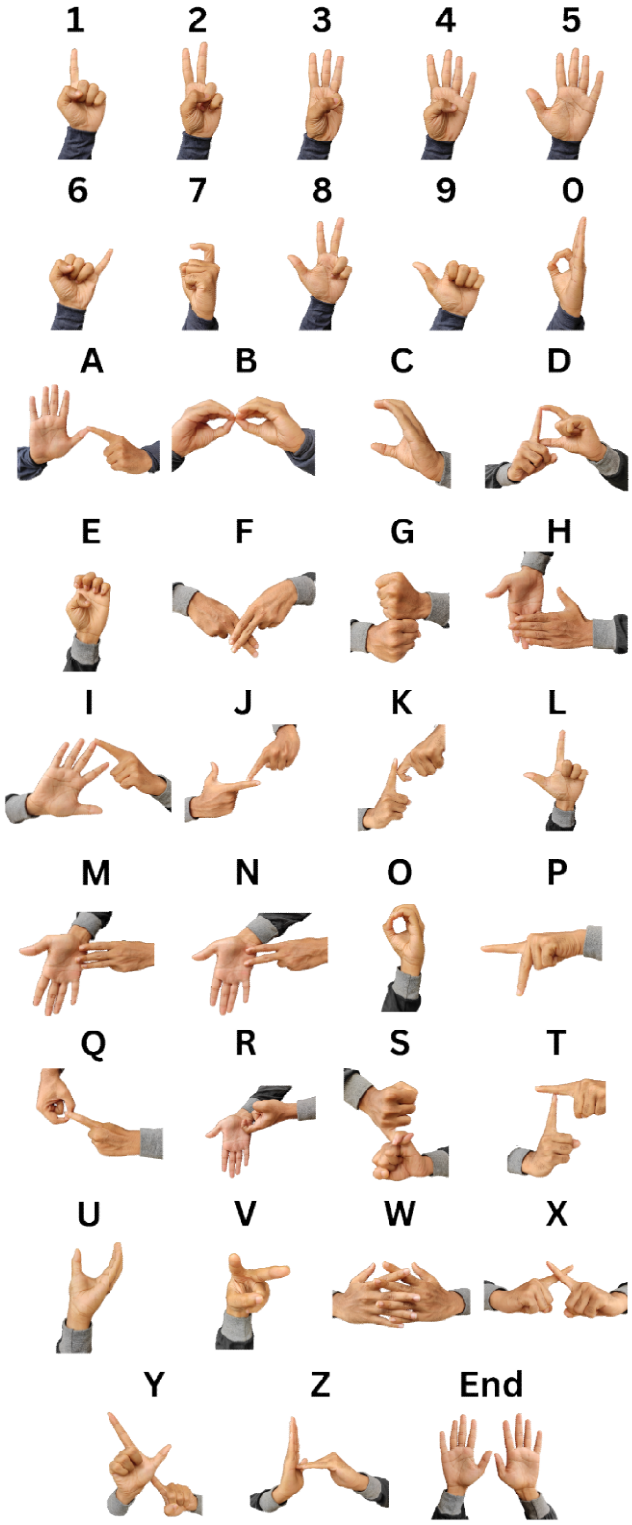

Fig 3.3. ISL signs for numbers and alphabet

The collected data is then put into two separate comma separated values (csv files) named keypoint.csv and keypoint_classifier_label.csv :

Keypoint.csv: This file contains all the knuckle points fetched by mediapipe. Each point has x and y coordinates which are separately placed. For two hands there are 2 * (21 * 2) = 84 data points along with the gesture index. So, keypoint.csv file contains a total of 85 data points.

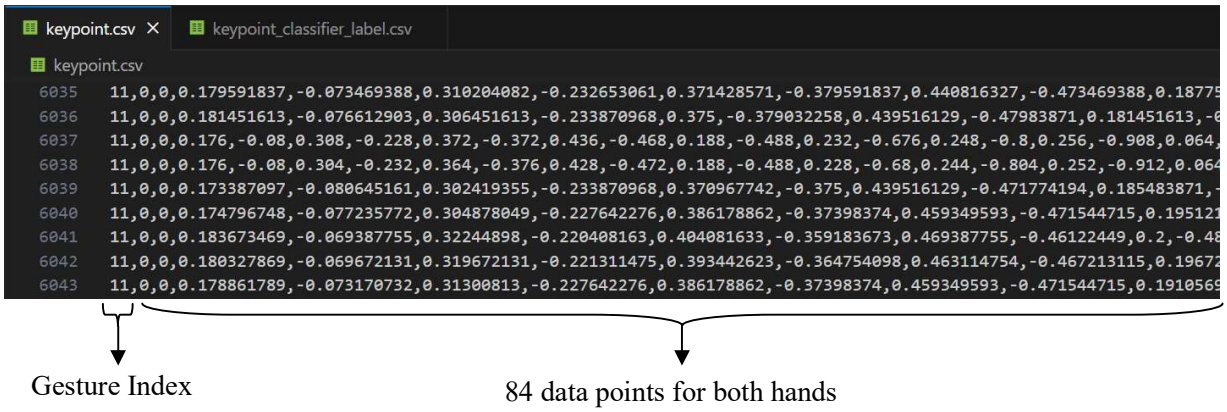

Fig 3.4. keypoint.csv file

keypoint_classifier_label.csv: This file contains the hand gesture value corresponding to each hand gesture id stored in the keypoint.csv file. The contents of this file are taken as direct input from the user. While capturing images first the user is asked to enter the label value for the image or sign that he/she is going to record. The value entered by the user is stored as it is in the keypoint_classifier_label.csv file.

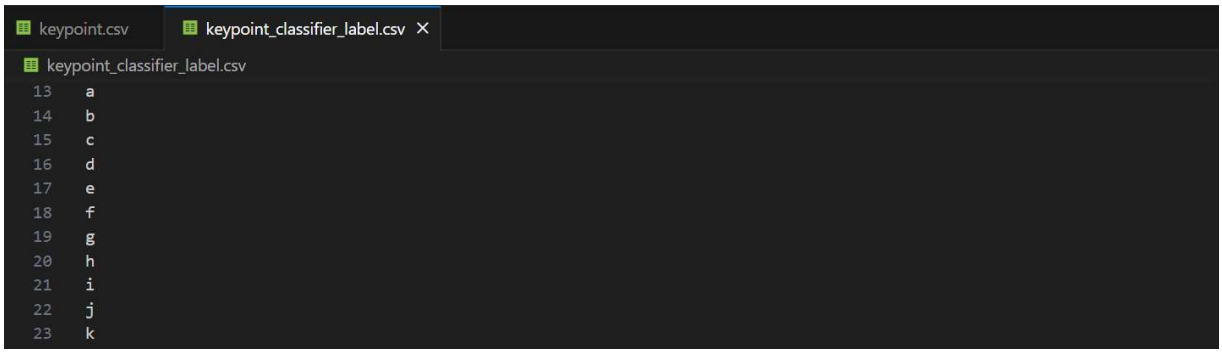

Fig 3.5. keypoint_classifier_label.csv file

3.2. Designing

The fundamental idea underpinning the design of any distributed system is system design. System design is the process of developing an architecture for various system modules, interfaces, and components as well as supplying pertinent data that is used when putting those components into practice.

3.2.1. Design Overview

We initially require photos for model training. Numerous functions are available in the Python opencv package to retrieve and manipulate webcam images and videos. Frames from the user's video feed were captured using Opencv. Then, these frames were sent to Mediapipe, a library that scans hand landmarks and outputs the x and y coordinates of the significant hand landmarks. The labels for these key points will be obtained from the user and kept in a separate csv file after these key points have been saved in the csv file. The deep neural network architecture is next trained using the keypoints csv file, accuracy is assessed, and the model is optimised to achieve the needed accuracy.

The mediapipe library's hand landmarks are retrieved and supplied into the model for inference. The received landmark is categorized by this model into one of the classes for which the data has been gathered. The model and the labels are queried for the gesture id. The relevant hand gesture name is extracted from the csv file. When the statement completes, the results are saved in a string, and the full string is then provided to the GingerIt package, which fixes the grammar and creates a proper sentence from the string. The Argostranslate program, which translates sentences from one language to another, is then given the sentence. We have adjusted the language to Hindi for ease of use. As a result, we will receive a final output statement in both Hindi and English.

Python GUI development makes use of Tkinter. To make it user-friendly and simple enough for anyone to use, we kept the user interface minimalistic.

The following page displays a diagrammatic depiction of the full model.

3.2.2. System Representation

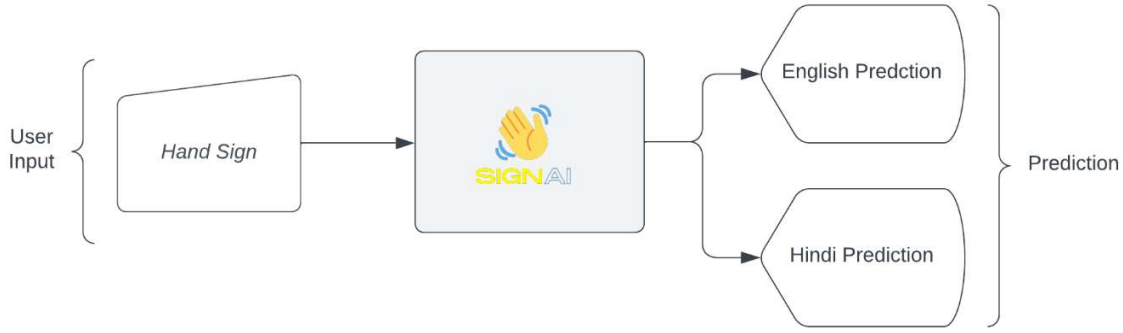

Fig 3.1. Context Diagram

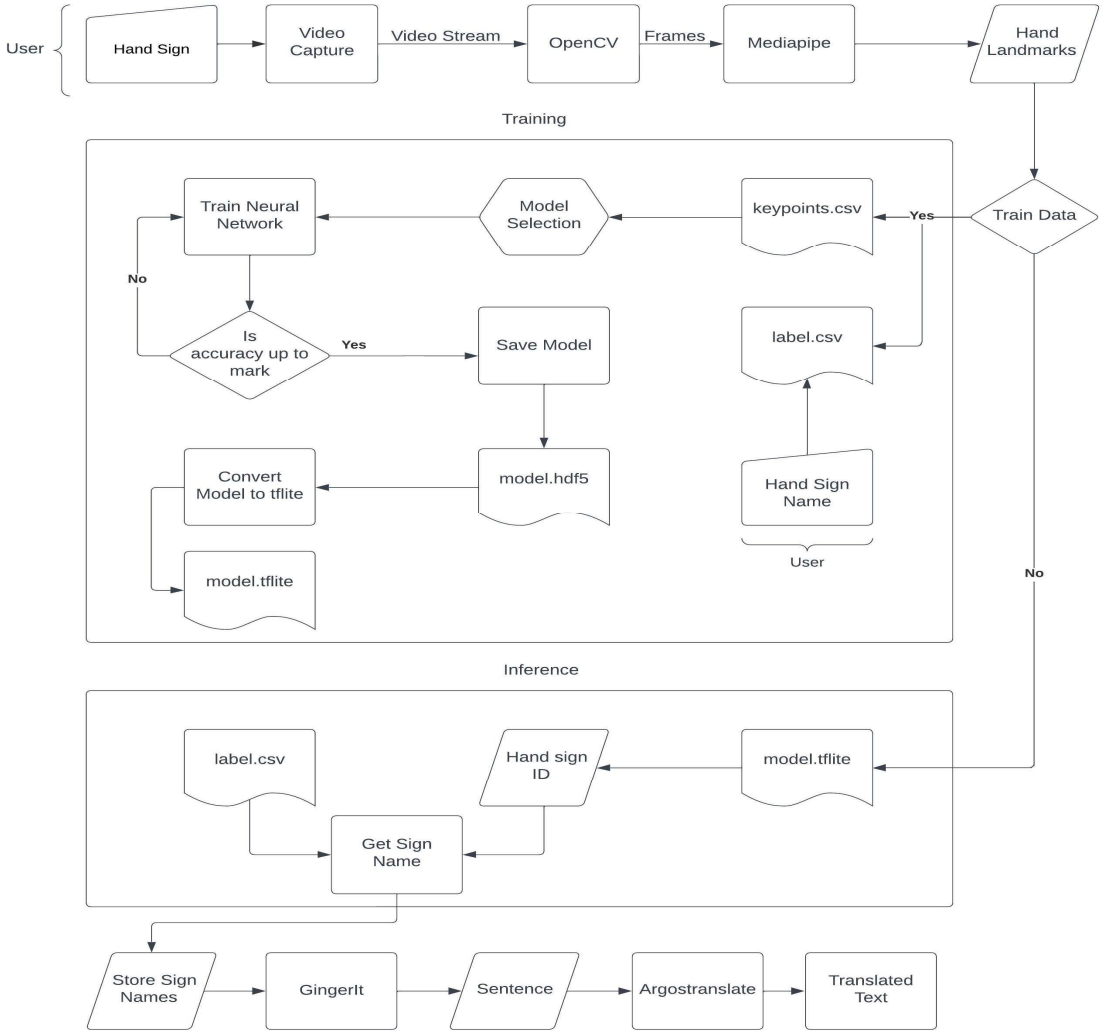

Fig 3.2. System Representation

3.2.3. Key Technologies

The following key technologies are used to develop the whole model:

- **OpenCV**

  OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and licensed as free and open-source software under Apache License 2. Starting in 2011, OpenCV features GPU acceleration for real-time operations. OpenCV is written in the programming language C++, as is its primary interface, but it still retains a less comprehensive though extensive older C interface. All newer developments and algorithms appear in the C++ interface. There are language bindings in Python, Java, and MATLAB/Octave.

- **MediaPipe**

  MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works on Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano. Since 2012, Google has used it internally in several products and services. It was initially developed for real-time analysis of video and audio on YouTube. Gradually it got integrated into many more products. MediaPipe requires minimal resources. It is so tiny and efficient that even embedded IoT devices can run it.

- **Deep Learning**

  Deep learning is part of a broader family of machine learning methods, which is based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks, convolutional neural networks and transformers have been applied to fields including computer vision, speech recognition, natural language processing, machine

translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

- **Neural Networks**

  A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

- **Tensorflow**

  TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

- **Keras**

  Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating

System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the Xception deep neural network model.

- **Tensorflow Lite**

  TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices. A TensorFlow Lite model is represented in a special efficient portable format known as FlatBuffers (identified by the *.tflite* file extension). This provides several advantages over TensorFlow's protocol buffer model format such as reduced size (small code footprint) and faster inference (data is directly accessed without an extra parsing/unpacking step) that enables TensorFlow Lite to execute efficiently on devices with limited compute and memory resources. A TensorFlow Lite model can optionally include *metadata* that has human-readable model description and machine-readable data.

- **Tkinter**

  Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface. Tkinter was written by Steen Lumholt and Guido van Rossum, then later revised by Fredrik Lundh. Tkinter is free software released under a Python license.

- **GingerIt**

  Used For correcting spelling and grammar mistakes based on the context of complete sentences. Wrapper around the gingersoftware.com API. In the project this package is used to make grammatically correct sentences out of the text generated by the project by reading the hand gestures.

- **Argostranslate**

  Open-source offline translation library written in Python. Uses OpenNMT for translations, SentencePiece for tokenization, Stanza for sentence boundary recognition, and PyQt for GUI. Designed to be used as either a Python library, command-line, or GUI application.

- **Computer Vision**

  Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions.

- **Natural Language Processing**

  Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

- **Python**

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

3.3. File Structure

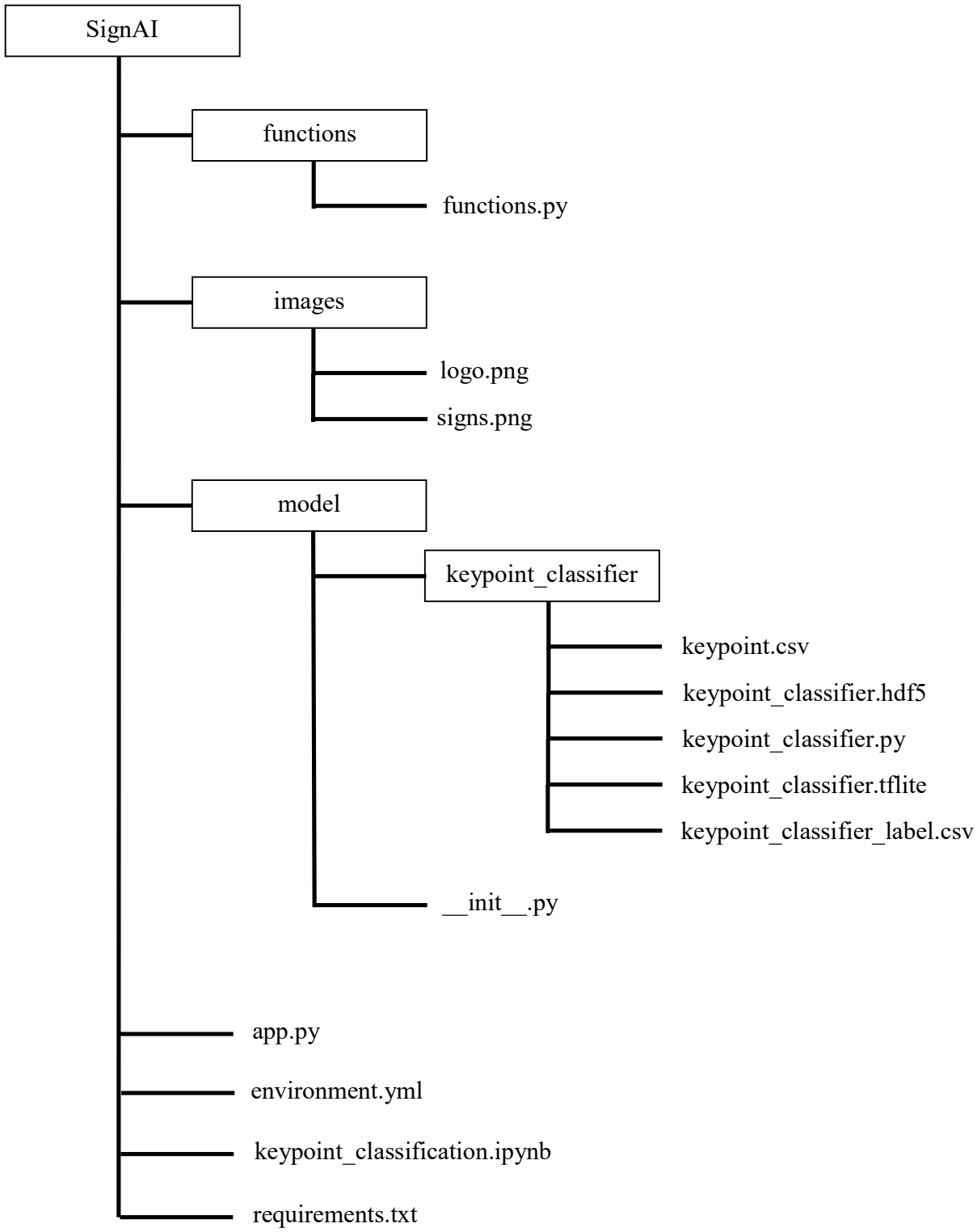

Fig 3.3. File Structure

Chapter 4

Implementation

4. IMPLEMENTATION

4.1. Model Building

After completing the whole design phase now we have a concrete idea of the overall workflow of the project and we are ready to implement it. The model will be made using neural network sequential model of keras which will be discussed in detail below.

4.1.1. Introduction to neural networks

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

4.1.2. Perceptrons

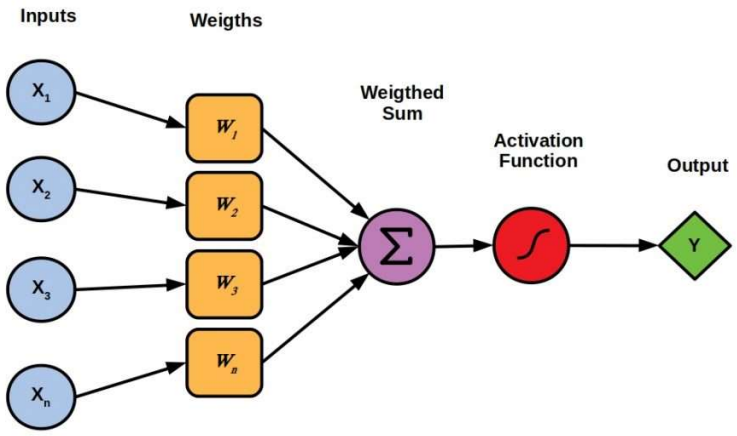

Fig 4.1. Perceptron

Perceptrons — invented by Frank Rosenblatt in 1958, are the simplest neural network that consists of *n* number of inputs, only one neuron, and one output, where *n* is the number of

features of our dataset. The process of passing the data through the neural network is known as forward propagation and the forward propagation carried out in a perceptron is explained in the following three steps.

Step 1: For each input, multiply the input value $\mathbf{x_i}$ with weights $\mathbf{w_i}$ and sum all the multiplied values. Weights — represent the strength of the connection between neurons and decides how much influence the given input will have on the neuron's output. If the weight w_1 has a higher value than the weight w_2, then the input x_1 will have a higher influence on the output than w_2.

$$\sum = (x_1 \times w_1) + (x_2 \times w_2) + \cdots + (x_n \times w_n)$$

The row vectors of the inputs and weights are $x = [x_1, x_2, \ldots, x_n]$ and $w = [w_1, w_2, \ldots, w_n]$ respectively and their *dot product* is given by:

$$x.w = (x_1 \times w_1) + (x_2 \times w_2) + \cdots + (x_n \times w_n)$$

Hence, the summation is equal to the *dot product* of the vectors *x* and *w*

$$\sum = x.w$$

Step 2: Add bias **b** to the summation of multiplied values and let's call this *z*. Bias — also known as the offset is necessary in most of the cases, to move the entire activation function to the left or right to generate the required output values.

$$z = x.w + b$$

Step 3: Pass the value of *z* to a non-linear activation function. Activation functions — are used to introduce non-linearity into the output of the neurons, without which the neural network will just be a linear function. Moreover, they have a significant impact on the learning speed of the neural

network. Perceptrons have *binary step function* as their activation function. However, we shall use s*igmoid* — also known as *logistic* function as our activation function.

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

4.1.3. Learning Algorithm

The learning algorithm consists of two parts — backpropagation and optimization.

Backpropagation: Backpropagation, short for *backward propagation of errors*, refers to the algorithm for computing the gradient of the loss function with respect to the weights. However, the term is often used to refer to the entire learning algorithm. The backpropagation carried out in a perceptron is explained in the following two steps.

Step 1: To know an estimation of how far are we from our desired solution a *loss function* is used. Generally, *mean squared error* is chosen as the loss function for regression problems and *cross entropy* for classification problems. Let's take a regression problem and its loss function be mean squared error, which squares the difference between *actual* (y_i) and *predicted value* ($\hat{y}_i$).

$$MSE_i = (y_i - \hat{y}_i)^2$$

Loss function is calculated for the entire training dataset and their average is called the *Cost function* **C**.

$$C = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Step 2: In order to find the best weights and bias for our Perceptron, we need to know how the cost function changes in relation to weights and bias. This is done with the help of the *gradient*s (*rate of change)* — how one quantity changes in relation to another quantity. In our case, we need to find the gradient of the cost function with respect to the weights and bias.

Let's calculate the gradient of cost function **C** with respect to the weight $\mathbf{w_i}$ using *partial derivation.* Since the cost function is not directly related to the weight w_i, let's use the chain rule.

$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_i}$$

Now we need to find the following three gradients

$$\frac{\partial C}{\partial \hat{y}} = ? \qquad \frac{\partial \hat{y}}{\partial z} = ? \qquad \frac{\partial z}{\partial w_1} = ?$$

Let's start with the gradient of the c*ost function (C)* with respect to the *predicted value* (ŷ)

$$\frac{\partial C}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 2 \times \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)$$

Let $y = [y_1 , y_2 , \dots y_n]$ and $\hat{y} = [\hat{y}_1 , \hat{y}_2 , \dots \hat{y}_n]$ be the row vectors of actual and predicted values. Hence the above equation is simplified as

$$\frac{\partial C}{\partial \hat{y}} = \frac{2}{n} \times sum(y - \hat{y})$$

Now let's find the gradient of the *predicted value* with respect to the *z*. This will be a bit lengthy.

$$\begin{aligned}
\frac{\partial \hat{y}}{\partial z} &= \frac{\partial}{\partial z}\sigma(z) \\
&= \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1}{(1+e^{-z})} \times \frac{e^{-z}}{(1+e^{-z})} \\
&= \frac{1}{(1+e^{-z})} \times \left(1 - \frac{1}{(1+e^{-z})}\right) \\
&= \sigma(z) \times (1-\sigma(z))
\end{aligned}$$

The gradient of z *with respect to the weight* $\boldsymbol{w_i}$ *is*

$$\begin{aligned}
\frac{\partial z}{\partial w_i} &= \frac{\partial}{\partial w_i}(z) \\
&= \frac{\partial}{\partial w_i} \sum_{i=1}^{n} (x_i.w_i + b) \\
&= x_i
\end{aligned}$$

Therefore we get,

$$\frac{\partial C}{\partial w_i} = \frac{2}{n} \times sum(y - \hat{y}) \times \sigma(z) \times (1-\sigma(z)) \times x_i$$

What about Bias? — Bias is theoretically considered to have an input of constant value *1*. Hence,

$$\frac{\partial C}{\partial b} = \frac{2}{n} \times sum(y - \hat{y}) \times \sigma(z) \times (1 - \sigma(z))$$

Optimization: Optimization is the selection of the best element from some set of available alternatives, which in our case, is the selection of best weights and bias of the perceptron. Let's choose *gradient descent* as our optimization algorithm, which changes the *weights* and *bias*, proportional to the negative of the gradient of the cost function with respect to the corresponding weight or bias. *Learning rate* (α) is a hyper parameter which is used to control how much the weights and bias are changed.

The weights and bias are updated as follows and the backpropagation and gradient descent is repeated until convergence.

$$w_i = w_i - \left(\alpha \times \frac{\partial C}{\partial w_i}\right)$$

$$b = b - \left(\alpha \times \frac{\partial C}{\partial b}\right)$$

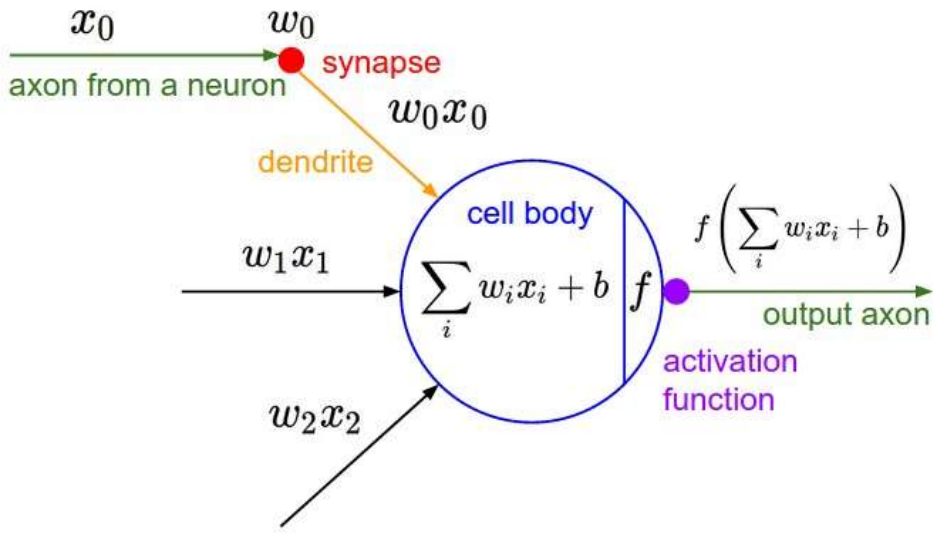

Fig 4.2. Perceptron Functions

4.1.4. Simple Neural Network Architecture

A basic neural network has interconnected artificial neurons in three layers:

Input Layer: Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.
Hidden Layer: Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
Output Layer: The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

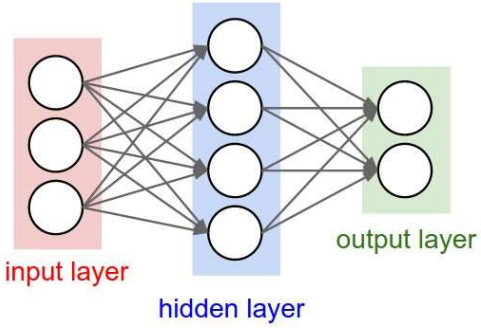

Fig 4.3. Two Layer Neural Network

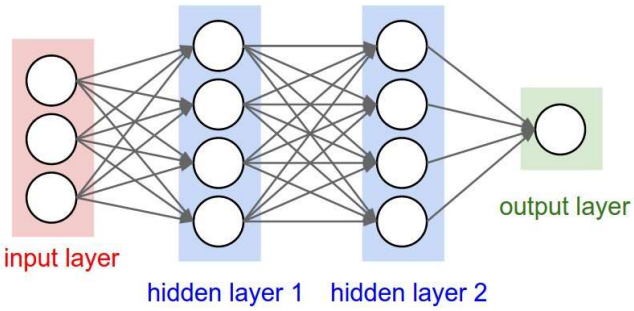

Fig 4.4. Three Layer Neural Network

4.1.5. Sequential Network Model

Keras model represents the actual neural network model. Keras provides two mode to create the model, simple and easy to use Sequential API as well as more flexible and advanced Functional API.

The core idea of Sequential API is simply arranging the Keras layers in a sequential order and so, it is called Sequential API. Most of the ANN also have layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((42 * 2, )),
    tf.keras.layers.Dense(42, activation='relu', name = "Layer_1"),
    tf.keras.layers.Dense(10, activation='relu', name = "Layer_2"),
    tf.keras.layers.Dense(20, activation='relu', name = "Layer_3"),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax', name = "Output")
])
```

Fig 4.5. Creating Sequential Neural Network

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 42)                3570

 Layer_2 (Dense)             (None, 10)                430

 Layer_3 (Dense)             (None, 20)                220

 Output (Dense)              (None, 39)                819

=================================================================
Total params: 5,039
Trainable params: 5,039
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.6. Model Summary

The model consists of 4 layer excluding one input layer. The input layer has 84 units of perceptron which represent all 84 keypoints of the hand landmark stored in keypoints.csv file. Output from each layer is accepted by the next layer as input. The first layer takes 84 input values and outputs 42 values. The second layer takes 42 input values and sends 10 outputs to next layer. The third layer receives 10 inputs and generates 20 output. The fourth and output layer takes those 20 values as input and sends output equal to number of classes of signs we have till now stored.

- **Unit**

  A Unit is a single neuron or perceptron that has its own activation function, weight and bias. Using these three parameters, it applies transformation on received input and generates output that is sent to the next layer.

- **Layer**

  A Layer is a group of units that perform same type of job in unison. In other words, a layer is a group of neurons that receive the same class of input from the previous layer and generate same part of output and sends it to the next layer.

- **Parameters**

  Each neuron in a dense layer has its own set of weights and a bias terms. These weights and biases are the parameters that are learned during the training process to make the network adapt to the given task. During the forward pass of training or inference, the inputs are multiplied by the weights, and the biases are added to compute the output of each neuron in the dense layer. The output then becomes the input for the next layer in the network.

- **Dense Layer**

  A dense layer, also known as a fully connected layer, is one of the fundamental building blocks in a neural network. It is called "dense" because every neuron in the current layer is connected to every neuron in the previous layer. In a dense layer, each neuron receives

input from all neurons in the previous layer and applies a linear transformation followed by an activation function to produce an output. The transformation includes multiplying the input values by learnable weights and adding a bias term.

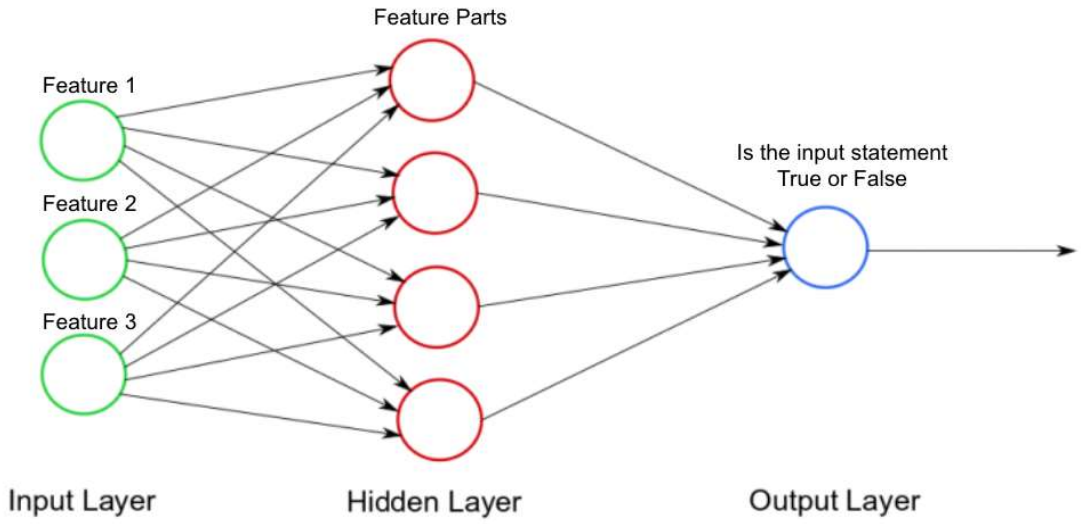

Fig 4.7. Dense Neural Network

- **Activation**

  An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

  The two activation function that are used while creating the model is ReLU activation function and Softmax activation function.

- **ReLU**

  ReLU stands for Rectified Linear Unit. Although it gives an impression of a linear function, ReLU has a derivative function and allows for back propagation while simultaneously making it computationally efficient. The main catch here is that the ReLU

function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0.

Mathematically it can be represented as:

$$f(x) = max\ (0, x)$$

The advantages of using ReLU as an activation function are as follows:
Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions. ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

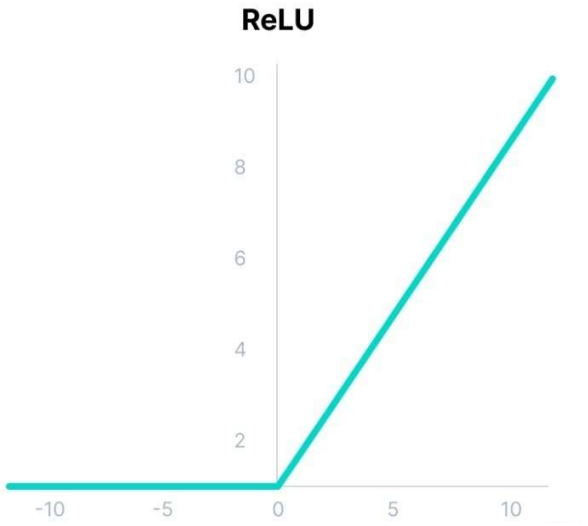

Fig 4.8. ReLU Activation Function

- **Softmax**

The output of the sigmoid function was in the range of 0 to 1, which can be thought of as probability. But this function faces certain problems. Let's suppose we have five output values of 0.8, 0.9, 0.7, 0.8, and 0.6, respectively. How can we move forward with it? The answer is: We can't. The above values don't make sense as the sum of all the classes/output probabilities should be equal to 1. You see, the Softmax function is

described as a combination of multiple sigmoids. It calculates the relative probabilities. Similar to the sigmoid/logistic activation function, the SoftMax function returns the probability of each class. It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification.

Mathematically it can be represented as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

4.2. Model Evaluation

While creating data for the system the data was divided in 8.5:1.5 ratio for training and testing i.e. 85% of the data was reserved for training the model and 15% data was hidden for testing the model after the model has been trained.

Model evaluation generates two results accuracy and loss. Accuracy is the measure of how accurately the model predicts the result. The predicted result is matched with the predefined labels to check what are the instances in which the model predicts correctly. The loss is the measure of how often the model fails to predict the result correctly. While evaluating a model we try to maximize the accuracy. Accuracy of the model is directly proportional to better prediction. Minimizing loss is also a property of good model training.

In our model we got accuracy of 0.9987 and loss of 0.0219.

```
# Model evaluation
val_loss, val_acc = model.evaluate(X_test, y_test, batch_size=128)
```

```
80/80 [==============================] - 0s 2ms/step - loss: 0.0219 - accuracy: 0.9987
```

Fig 4.9. Evaluation Values

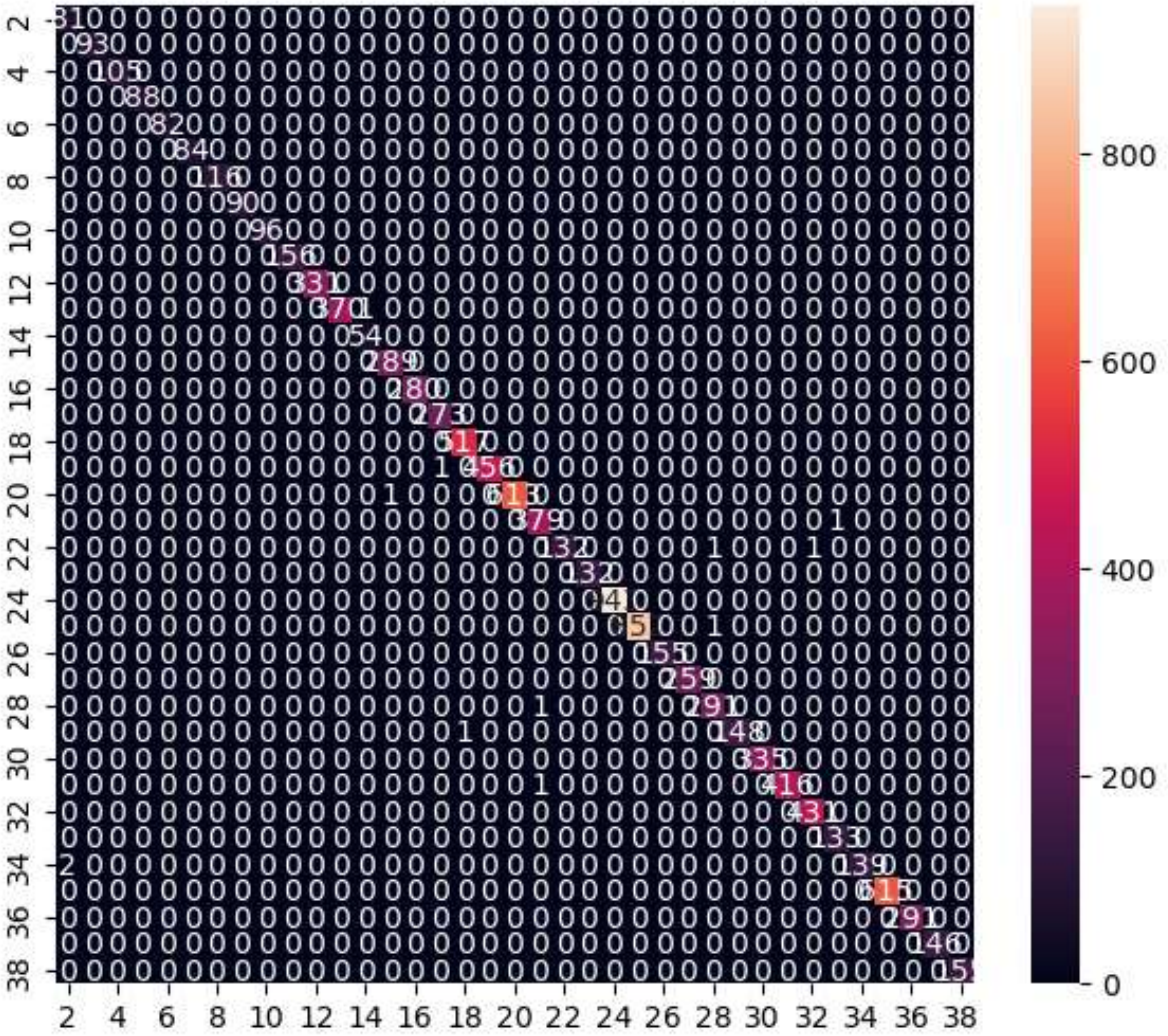

Fig 4.10. Confusion Matrix

A confusion matrix is a table that is often used to evaluate the performance of a classification model. It summarizes the predictions made by the model on a set of test data and compares them to the actual ground truth labels. It provides a detailed breakdown of the model's performance by showing the counts of true positive, true negative, false positive, and false negative predictions typically a supervised learning algorithm; in unsupervised learning it is usually called a matching matrix.

4.3. Graphical User Interface

The graphical user interface, or GUI is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based UIs, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

Important qualities of User Interface Design are following :

1. **Simplicity :**
   - Less number of mouse clicks and keystrokes are required to accomplish this task.
   - It is important that new features only be added if there is compelling need for them and they add significant values to the application.
2. **Consistency :**
   - Consistency also prevents online designers information chaos, ambiguity and instability.
   - We should apply typeface, style and size convention in a consistent manner to all screen components that will add screen learning and improve screen readability. In this we can provide permanent objects as unchanging reference points around which the user can navigate.
3. **Intuitiveness :**
   - Intuitive user interface design is one that is easy to learn so that user can pick it up quickly and easily.
   - Icons and labels should be concise and cogent. A clear unambiguous icon can help to make user interface intuitive and a good practice is make labels conform to the terminology that the application supports.
4. **Prevention :**
   - A good user interface design should prevents users from performing an in-appropriate task and this is accomplished by disabling or “graying cut” certain elements under certain conditions.

5. **Forgiveness :**

- This quality can encourage users to use the software in a full extent.
- Designers should provide users with a way out when users find themselves somewhere they should not go.

The graphical user interface for the project is developed in tkinter and all the necessary components needed for smooth running of the application has been included. The interface is simple, easy to use and very user friendly.

The user interface of the project is shown below:

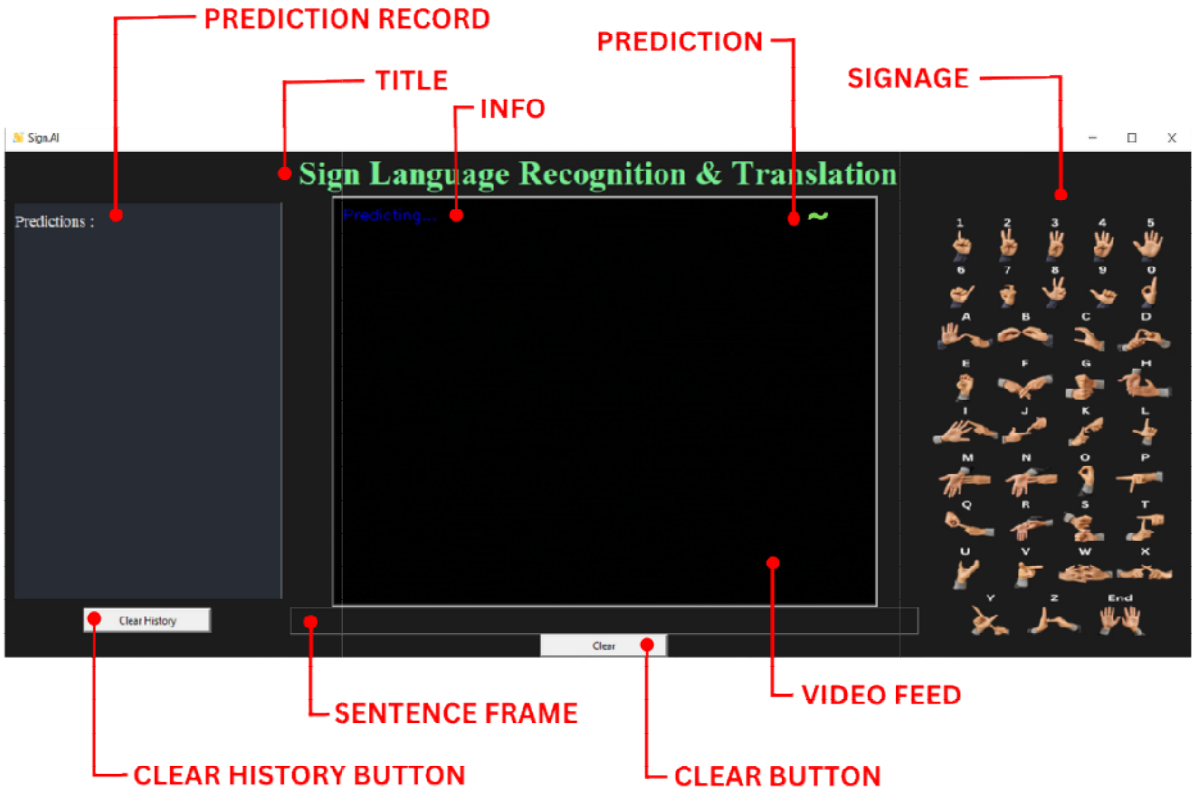

Fig 4.11. Labeled User Interface

The user interface has following components:

- **Title**

  This displays the title of the Project “Sign.AI”.

- **Video Feed**

  This is the frame where the users shows the signs and the frames are captured and processed to derive the meaning of those signs. Using mediapipe the hand skeleton is also displayed which reads the hand gesture and returns the hand landmarks.

- **Info**

  Info shows information or status of the current feed. If the model is under training then it shows the gesture id and the number of frames captured. If it is predicting then it simply shows the text “Predicting…”.

- **Prediction**

  The predictions for the current sign is displayed here. These predictions are then combined later to form sentences.

- **Sentence Frame**

  The sentence frame collects each of the predictions made by the model sequentially. This does not modify the content in any way. It just stored the exact predictions shown in the video feed by the user.

- **Clear Button**

  The clear button clears the sentence frame. The sentence frame is automatically cleared after recognizing the “fullstop” or “end” sign. In case of any wrong entries user can clear the frame manually.

- **Signage**

  This is the chart showing all the symbol for which the model has been trained and their

corresponding labels for the ease of user.

- **Prediction Record**

  The Prediction Record contains the grammatically correct sentences that have been recorded by the user. It also contains Hindi translation of those sentences. It is a scrollable frame so it is capable of storing a large generated text data.

- **Clear History Button**

  The Clear History button clears the history palette.

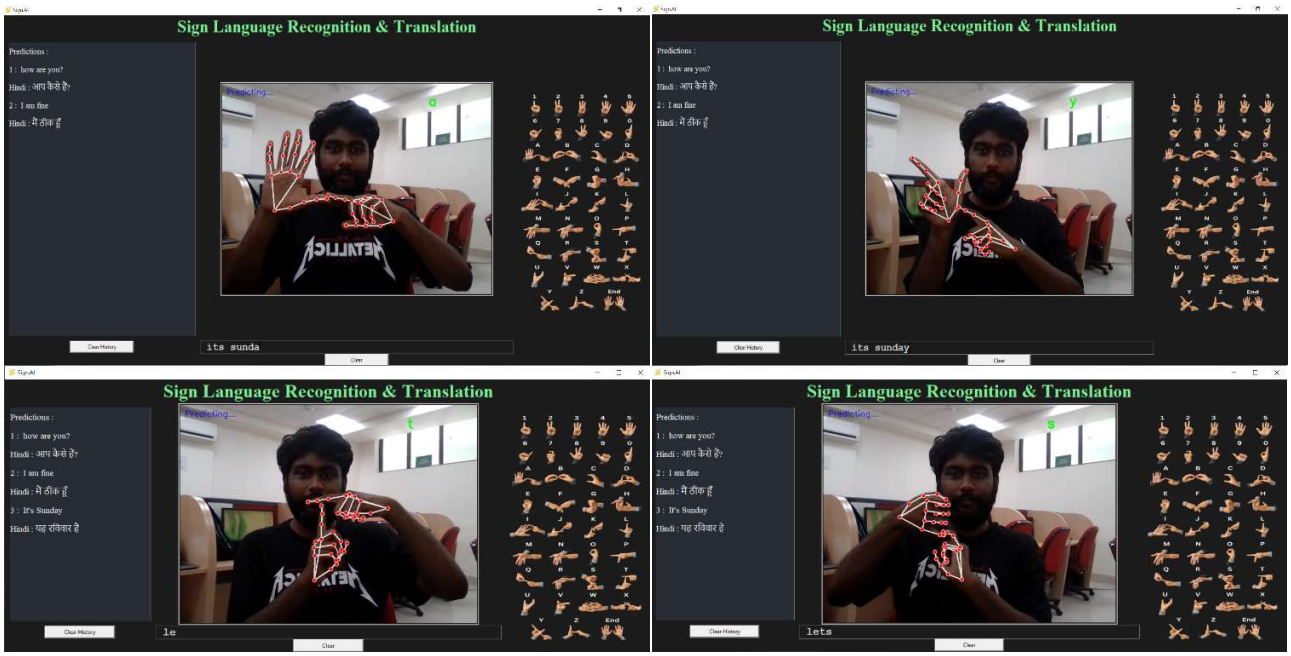

Fig 4.12. GUI Inference

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

5.2 Future Scope

5. CONCLUSION AND FUTURE SCOPE

5.1. Conclusion

In conclusion, the project on **Sign Language Recognition and Translation** is fully capable to successfully address the significant challenge of bridging the communication gap between individuals with hearing impairments and the wider society. Through the use of advanced technologies, including computer vision and deep learning, we have developed an effective system for detecting and interpreting gestures in real time.

The project involved several key stages, starting with the collection and preprocessing of a comprehensive dataset. This dataset served as the foundation for training our deep learning models, allowing them to recognize and classify a wide range of gestures accurately. For training the digits and the letter we required more than 50000 images to train the model with wide variety of hand frames. The evaluation of our system involved rigorous testing and validation against a diverse set of sign gestures. Several phases of testing and training were involved to increase the accuracy of the model to the required values so that it becomes fully capable of detecting hand gestured in real time. The results demonstrated a high level of accuracy and reliability, with our models consistently achieving over 90% accuracy in gesture recognition. It is important to note that this project is just a stepping stone towards a more inclusive and accessible society. While our system is a valuable tool for sign recognition, further research and development are needed to refine its capabilities, expand its vocabulary, and accommodate regional variations. Additionally; efforts should be made to make the system more accessible through user-friendly interfaces and integration with existing assistive technologies.

5.2. Future Scope

While the Indian Sign Language (ISL) detection project has made significant strides in bridging the communication gap, there are several areas for further improvement and future development.

- This project can be implemented in various platforms like android and iOS, increasing the reach of the application allowing people to access it easily through their mobile devices.
- The graphical user interface can be improved so that the UI becomes more user friendly making it simple to use.
- More user options can be provided so that user can have access to various functionalities of the application such as saving and sharing the generated file, choosing the language into which the user needs the output to be displayed.
- Speech output can be generated so that user will not have to read each text and its more convenient to hear to the recognized gesture.
- There is always scope for improvement in the model to make the project a better version.
- Hardware improvements can enhance the performance of the project to a great extent, use of good camera helps mediapipe to track the hands more accurately, usage of high performance CPU and GPU trains the model much faster hence saves training time.
- Giving user the functionality to train his own gestures, till now we have not provided the training functionality to the user to keep the application simple to use.
- Scalability can be improved by training the model with all the symbols used in the ISL vocabulary. The current model is only trained with numbers 0 to 9 and alphabets. Training more symbols will aid in large scale use of the project.
- Using better sentence generation and translation APIs to generate better result.
- Current model required internet connection for the translation of text into Hindi. Offline translation can be implemented for better adaptability in areas with low network connectivity.

These are some of the improvements we could think of. We are working on these ourselves but these are not part of the project at current stage.

BIBLIOGRAPHY

1. TensorFlow API Documentation: TensorFlow. (n.d.). TensorFlow API documentation. Retrieved from https://www.tensorflow.org/api_docs/python/tf
2. TensorFlow Lite API Documentation: TensorFlow. (n.d.). TensorFlow Lite API documentation. Retrieved from https://www.tensorflow.org/lite/api_docs
3. Indian Sign Language: Indian Sign Language. (n.d.). Official website. Retrieved from https://indiansignlanguage.org/
4. OpenCV Documentation: OpenCV. (n.d.). OpenCV documentation. Retrieved from https://docs.opencv.org/4.x/
5. Stack Overflow: Stack Overflow. (n.d.). Stack Overflow website. Retrieved from https://stackoverflow.com/
6. Wikipedia: Wikipedia. (n.d.). Wikipedia website. Retrieved from https://www.wikipedia.org/
7. PyPI (Python Package Index): Python Package Index. (n.d.). PyPI website. Retrieved from https://pypi.org/
8. MediaPipe Documentation: Google. (n.d.). MediaPipe documentation. Retrieved from https://developers.google.com/mediapipe/
9. Argos Translate Documentation: Argos Translate. (n.d.). Argos Translate documentation. Retrieved from https://argos-translate.readthedocs.io/en/latest/
10. MediaPipe by Google: Google. (n.d.). MediaPipe by Google. Retrieved from https://google.github.io/mediapipe/

ABBREVIATIONS & ACRONYMS

ISL – Indian Sign Language

AI – Aritificial Intelligence

CPU – Central Processing Unit

RAM – Random Access Memory

MP – Megapixels

FPS – Frames Per Second

MB – MegaByte

GPU – Graphic Processing Unit

VS – Visual Studio

CSV – Comma Separated Values

TFLITE – Tensorflow lite

TF – Tensorflow

GUI – Graphical User Interface

CV – Computer Vision

NLP – Natural Language Processing

MSE – Mean Squared Error

API – Application Programmer Interface

ReLU – Rectified Linear Unit

EXP – Exponent

CLI – Command Line Interface

UI – User Interface

NN – Neural Networks

CNN – Convolutional Neural Networks