MIT | Academy of Engineering
(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

A Project Report on

Sign Language to Audio Converter

Submitted by,

Chatak Shinde	(Exam Seat No. B227004)
Pranav Unkule	(Exam Seat No. B227049)
Pratik Saurkar	(Exam Seat No. B227002)
Sanchit Agarkar	(Exam Seat No. B227033)

Guided by,

Dr. Usha Verma

A Report submitted to MIT Academy of Engineering, Alandi(D), Pune, An Autonomous Institute Affiliated to Savitribai Phule Pune University in partial fulfillment of the requirements of

BACHELOR OF TECHNOLOGY in
Electronics & Telecommunication Engg.

School of Electrical Engineering
MIT Academy of Engineering
(An Autonomous Institute Affiliated to Savitribai Phule Pune University)
Alandi (D), Pune – 412105

(2022–2023)

MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

CERTIFICATE

It is hereby certified that the work which is being presented in the BTECH Project Report entitled **"Sign Language to Audio Converter"**, in partial fulfillment of the requirements for the award of the Bachelor of Technology in Electronics & Telecommunication Engg.and submitted to the **School of Electrical Engineering** of **MIT Academy of Engineering, Alandi(D), Pune, Affiliated to Savitribai Phule Pune University (SPPU), Pune**, is an authentic record of work carried out during Academic Year **2022–2023**, under the supervision of **Dr. Usha Verma**, **School of Electrical Engineering**

Chatak Shinde	**(Exam Seat No. B227004)**
Pranav Unkule	**(Exam Seat No. B227049)**
Pratik Saurkar	**(Exam Seat No. B227002)**
Sanchit Agarkar	**(Exam Seat No. B227033)**

Dr. Usha Verma	**Shridhar A. Khandekar**	**Dr. Dipti Y. Sakhare**
Project Advisor	**Project Coordinator**	**Dean SEE**

Director/Dy. Director(AR)　　**External Examiner**

DECLARATION

We the undersigned solemnly declare that the project report is based on our own work carried out during the course of our study under the supervision of **Dr. Usha Verma**.

We assert the statements made and conclusions drawn are an outcome of our project work. We further certify that

1. The work contained in the report is original and has been done by us under the general supervision of our supervisor.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this Institute/University or any other Institute/University of India or abroad.
3. We have followed the guidelines provided by the Institute in writing the report.
4. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Chatak Shinde	**(Exam Seat No. B227004)**
Pranav Unkule	**(Exam Seat No. B227049)**
Pratik Saurkar	**(Exam Seat No. B227002)**
Sanchit Agarkar	**(Exam Seat No. B227033)**

Abstract

Indian Sign Language (ISL) is widely utilised in India's deaf community. This project's primary objective is to develop a system that can interpret ISL in the areas of numerals, alphabets, and basic phrases, allowing the less fortunate people to communicate with the outside world without the use of an interpreter in public locations like banks and train terminals.The ability of sign language to bridge communication gaps is unknown to many parents of deaf children.Indian Sign Language (ISL) translators are in high demand in educational institutions and other areas where communication occurs. However, India has just about 300 licensed translators. So, here is the need to come forward with the concept of sign language to audio converter.

Numerous researchers have made efforts in this regard, but the primary problem with their method is that they only identified an alphabet or number with no additional output. As a result, work is done in the project not only to recognise the alphabets and numbers, but also to recast them into phrases and to convert those identified words into audio format.

The proposed Sign Language to Audio Converter system bridge the communication gap between speech impairment people and non-signers by taking video input from the camera. AlexNet and Long-Short-Term Mermory (LSTM) are the two deep learning algorithm used for classifying the real-time hand gestures.The Alex-Net base model focuses on classifying the static gestures and on the other hand Mediapipe with LSTM focuses on classifying the dynamic gestures.In case of AlexNet the dataset is created using image canny edge pre-processing technique. For the Mediapipe approach, key points are extracted and stored into the numpy format. After the prediction of gesture the predicted gesture is converted into audio.The AlexNet model had achieved the maximum accuracy of 97.51% with 90:10 traning-testing ratio and Mediapipe had achieved 80% with 95:5 traning-testing ratio.

The future goal of this project is to deploy a web application that will collect hand gestures through the camera, concatenate those detected movements to make a phrase, and output the audio response through the mobile speaker.

Acknowledgment

We want to express our gratitude towards our respected project advisor/guide Dr Usha Verma for her constant encouragement and valuable guidance during the completion of this project work. We also want to express our gratitude to respected School Dean Dr. Dipti Sakhare for her continuous encouragement.

We would be failing in our duty if we do not thank all the other staff and faculty members for their experienced advice and evergreen co-operation.

Chatak Shinde

Pranav Unkule

Pratik Saurkar

Sanchit Agarkar

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

People with hearing and speech impairments use sign language to communicate. Nonverbal communication, such as sign language gestures, is being used by people to convey their opinions and emotions. Non-signers, on the other hand, have a hard time understanding it, necessitating the use of expert sign language interpreters at legal and medical appointments, and also educational and training activities [1]. For individuals who can talk and listen, spoken language is a kind of communication. Many more oral languages exist in various countries across the world. But what about the deaf?

While automatic voice recognition is already commercially accessible, automatic Sign Language Recognition (SLR) is still in its infancy. All commercial translation services are now human-based, and hence expensive, due to the skilled staff necessary. As a result, the majority of public services are not translated into signs [3]. Because there is no widely used written sign language, all written communication is in the native spoken language. As a result, the notion of sign language to audio converter was developed.

Static and dynamic gesture recognition are the two types of gesture recognition. Static gesture recognition is a pattern recognition issue with a considerable pre-processing stage [6]. It is a very important stage in any traditional pattern recognition

Figure 1.1: Communication Gap

procedure. The characteristics are the most discriminative evidence about the image across classes. The head motions for ”No” and ”Yes” gestures in dynamic gestures can only be predicted with temporal context data. As deaf persons primarily focus on learning hand gestures for alphabets and numbers in order to connect with others, HGR using static hand gestures was chosen as the major topic of this research. Fig. 1.1 depicts the communication gap between deaf and hearing persons. Children in this illustration are unable to grasp what the deaf child is saying. The current work proposes a discriminative analysis of classes in order to determine the real finger-spelled letters of American Sign Language.

Individuals can speak with deaf and dumb persons because to technological advancements. Different machine learning models can be applied to create a system that bridges the communication gap between hearing and deaf persons. In Indian Sign Language, there are numerous gestures for different signs, sentences, alphabets, and numerals (ISL).

1.2 Motivation

Indian Sign Language (ISL) is widely utilised in India's deaf community. However, ISL is not utilized to teach deaf students in deaf schools. Teacher education programmers do not prepare instructors to employ ISL-based teaching strategies. There is no instructional material that includes sign language. Parents of deaf children are unaware of sign language's power to bridge communication gaps. [5]. ISL interpreters are urgently needed at institutes and places where deaf and hearing people communicate, yet India will only have less than 300 licenced interpreters by 2020. Sign languages have the issue of communicating meaning through several channels at the same time. While research into sign language linguistics is in in its early stages, it is clear that many of the techniques employed in speech detection are unsuitable for SLR [2]. Furthermore, the amount and quality of publicly available data sets are restricted, making many classic computer vision learning techniques insufficient for the task of developing classifiers.

Because of the constant drop in cost and size, computers have a bigger effect on people's daily lives as technology improves. Hand gestures have been utilised in communicating since the beginning of time. This type of recognition systems have found widespread use in medical applications, sign language interpretation and smart environments. Hand motions are primarily detected using three types of sensors: mount-based sensors, vision-based sensors and multi-touch screen sensors [4]. Because sensors which are based on vision have no physical connection with people, they are less awkward and more comfortable than mounted sensors. Fig. 1.2 Hand gesture recognition technology is displayed. In addition, vision-based sensors have a far greater operating range than multi-touch screen sensors.

The hand motions of Indian Sign Language (ISL) are shown in Fig. 1.3. This is how deaf people make meaningful words using the alphabet. In India, deaf individuals interact with other deaf people by using two hands. The alphabets from A to Z are illustrated here.

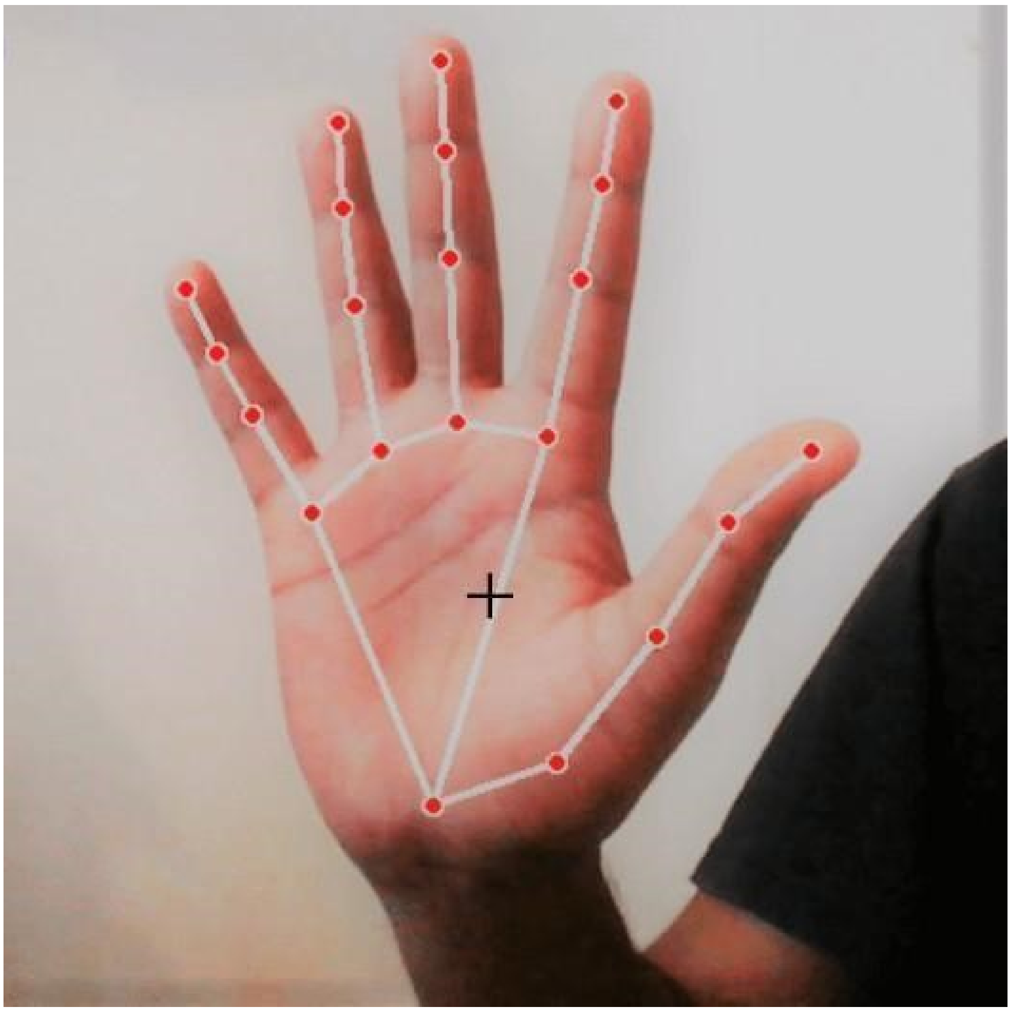


Figure 1.2: Relevant Technology

1.3 Project Challenges

Overfitting the data (using a model that is too intricate), underfitting the data (using a basic model), a lack of data, and non representative data are all issues in Machine Learning. Sign Language is vast. Background noise reduction was major challenge. If noise is presents in image then it may lead to false predication. which affects training accuracy of the model. To get higher accuracy dataset should be large. to learn large dataset higher computational power and large time required. This becomes computational barrier for training the model. To avoid this issue dataset should be more specific. Many hand gestures are dynamic gestures. Static gestures are easy to predict than dynamic gestures.

1.4 Proposed Solution

The capacity to solve difficulties is a fundamental life skill that is required in our daily lives. Issue solving is the process of detecting a problem, creating viable solutions, and deciding on the best course of action. Excellent problem-solving abilities empowers not just in your personal life, but also in your professional life.

The proposed solution is for specially able people which belongs to Indian community and used Indian Sign Language for their communication. the proposed model will recognise alphabets, numbers and words. It will recognise static as well as dynamic hand gestures. The recognised gestures were converted into text and further converted into an audio.

1.5 Project Report Organization

The report is subdivided into following sub-chapters:

Chapter 2: Literature Survey, reviews the contribution of researcher in the area of Sign language.At the end the previous work done by each researcher is summarized and the limitation and positive insights are used to propose the system.

Chapter 3: Problem Definition and Scope, this chapter consist problem statement for the topic and the semester wise objectives are defined.

Chapter 4: System Requirement Specification, this chapter deals with the user requirement and the hardware requirement that are required to withstand the system is explained.

Chapter 5: Methodology, this chapter contains the system architecture along with it mathematical modelling is also discussed for each and ever module such as pre-processing, algorithms.

Chapter 6: Implementation, this chapters constitutes of block diagram, flow of each and every step required for model to be build.Along with it the flow chart of dataset creation as well as flow chart for each AlexNet and Mediapipe based model is discussed.

Chapter 7: Result Analysis/Performance Evaluation, this chapter is subdivided into 3 main division starting with the results for each and every steps required for Alexnet model, similarly next for Mideapipe based approach and at last the comparative analysis is done.

Chapter 8: Conclusion, this chapter concludes the work by summarizing the results and the parameters.

Chapter 2

Literature Review

To develop this project many research articles were deeply studied from the literature survey. It is observed that some researchers preferred to work with Indian Sign Language(ISL) and some of them with American Sign Language(ASL). While studying the article it is observed that various techniques / algorithms used by researchers such as Principal Component Analysis(PCA), Convolution Neural Network(CNN) , K- nearest Algorithm etc. There are few of them who had fully-fledged detected sign language and converted it to audio format . Also it was observed that many of the projects are in development.

2.1 Related work

Recent work on sign language is studied and presented in this section. Ashish Sharma *et al* [1] have developed an Prominent techniques for feature extraction which are currently widely used for computer vision and recognition of gestures is developed so that they could be compared to the novel approach proposed in the paper. These techniques includes Histogram of Gradients(HOG), Principal Component Analysis(PCA) AND Local Binary Patterns(LBP). The main task of Histogram of Gradients as a feature descriptor is to take the image and extract all the essential zones out of it and then remove all the redundant ones left. Principal Component Analysis is a type of dimension reduction approach which includes Image Compression, Data Visualization, and Page Rank Algorithms. Local Binary Patterns acts as a texture operator which labels pixels by setting a threshold and approximating with the neighbouring pixels of the concerned pixels [1]. The size of Data-set used by the researcher is of 1GB (29 components each of 3000 images). In segmentation they have used canny edge detection to reproduce only the strong edges present in the image which helps in reducing the background noise.In feature extraction ORB feature detector is used to detect patches from the image and a 32-dimensional vector for each of the patches generated. Thus, for every image belonging to a set of a single class of sign images, a 32-dimensional vector of features is produced. So, key descriptors with identical features is produced and it is used here to generate a bag containing the feature models for all the images used for training. ORB is effective than HOG and PCA. So, ORB feature extraction is tested against other different preprocessing techniques such as HOG, PCA and LBP for the same dataset and the technique proposed here outperforms other pre-processing techniques for Logistic Regression, KNN classifiers and Naïve Bayes while Principal Component Analysis outperforms other techniques for Multi-Layer Perceptron, Random Forest and Support Vector Machine Classifiers.

Similarly, Kshitij Bantupalli *et al.* [2] also worked on the dataset of American Sign Language with one hundred different signs .In preprocessing CNN is used for two possible outputs. With the 2048-sized vector that the global pool layer provides, the RNN and LSTM may be able to classify more features utilising sequence data into one of the gesture classes.The two different approaches where used for the classification: a. utilising the predictions made by the Softmax layer and b. Using the global pool layer's output.The LSTM then classifies the gesture segments that the CNN recognised and processed.Because signers' faces differ, the model loses accuracy when faces are added because it ends up learning the wrong features from the videos [7].The model also performed poorly when the outfit was different. It might be more accurate to use a ROI to separate hand motions from the images, but for the purposes of this work, a full-sleeved shirt was worn consistently during all gesture recordings.Similary, Kohsheen Tiku *et al.* [3] also had propsed the work on American Sign Langugae(ASL) over which an android application is developed in android studio and open CV(java). The algorithm is developed on Java-based OpenCV wrapper. The researcher chose HOG because edge directions or intensity gradients make it simple to identify the shape and appearance of a nearby object. Three detection systems, including a detection method, a kernel, and a dimensionality reduction type, are included in the Support Vector Machine (SVM). ASL Kaggle dataset is used which has 3000(100)images for every alphabets.For feature extraction Downsampling, Grayscale conversion and Normalization is used.

For hand gesture recognition, Ms. Anamika Srivastava & Mr. Vikrant Malik [4] explained how Indian sign language(ISL) should be understood on a continuous basis. In this paper, the input image has been converted into binary pictures. The proposed work shows recognition of signs using Convolutional Neural Networks with 98 % accuracy. The dataset consists of two categories. which are, words of 5 letters and words of ten letters. PCANe (Principal Component Analysis Network) is used for feature extraction and a support vector machine is used for classifying the data. A public benchmark dataset and data collected from five users is used to test the system and it scored 88.7% average accuracy. In this paper, they proposed two approaches for classification first was using the pooling layers and another was using the softmax layer for final predictions. The softmax layer provides a better result cause of distinct features. For Data Augmentation cropping, scaling, rotating, and flipping techniques were used. The classification algorithm was Convolutional Neural Network using Inception v3 The obtained average validation accuracy is 90%(98 highest).

Comparably, in Sign Language Using Deep Neural Network: Images taken from Kaggle. DNN: Deep Neural Network is used for feature extraction. They use a three layer deep CNN for hand gesture recognition system. A Peak accuracy of 100 percent for the training and 82 percent for validation process is achieved. MOreover, a test accuracy of 70 percent is obtained for test images. The CNN model using Inception v3 has made astounding progress in the field of gesture recognition. With the average validation accuracy of 90 percent, and the greatest validation accuracy of 98 percent.

Shravani K. *et al* [5] have proposed a project which uses machine learning models for classification of Indian Sign languages. They have created a data of sign language for alphabets and numeric which contain 35 classes, each class with 1200 images, So total 42000 images.The steps involved are Image Collection, Image Processing, Feature Extraction. Bag of vision model is used to classify image. So, an image can be viewed as a document in order to depict any gesture using the BOW model. Canny Edge technique is used to detect the edge of an image. In Feature extraction, for feature detection sped up robust features (SURF) is used. The histogram is calculated by finding the frequency of occurrence of each visible word that belongs to the image in total visual word. They have divided the data in 80:20 ratio, such that each class has 960 images for training and 240 images for testing. A support Vector Machine is used to train and predict the data with a linear kernel.Additionally, real-time gesture prediction using video feed has been made possible thanks to real-time recognition technology. The Bow-integrated SURF feature descriptions gave the model a 99.9% accuracy rating. slight bias in the model's prediction due to the data set's extremely similar photos, which don't differ in things like light and skin tone. So, this approach can be more reliable for use in real-world applications by using a broad and diverse range of photos in the data set.

Similarly, Kusumika Krori Dutta [6] has explained that both single-handed and double-handed hand motions are used to communicate in Indian Sign Language. The topic of this paper is, the classification of Indian sign language using machine learning. In this paper, Principal Component Analysis (PCA) and an artificial neural network (ANN) technique were implemented in MATLAB and used to train the system using double-handed sign language. So, the database was created which includes different single-handed gesture images which were then reduced to 256X256 because for faster classification size reduction is crucial and the size reduction was done using PCA. The model was then trained using the K Nearest Neighbor Algorithm (k-NN) and the Back Propagation Algorithm. Basically, k-NN is used to classify the new data point into the target class and the Back Propagation Algorithm is used to update the weights of the network in order to reduce the error. Whenever a test sample is passed it goes through kNN and backpropagation algorithm to verify the input and gives output. Then the verified output goes under text-to-speech conversion and we get the final output in form of sound /speech. The accuracy for alphabet recognition is approx 94.88%.

On the other hand, Dasari Vishal *et al* [9] proposed designs that required bulky hardware such as a webcam and computer which led the system to be static. So to avoid these author came up with a design in which the whole system will be embedded on a wristband . Basically the wristband is compromises of 3 different types of sensors namely a gyroscope, accelerometer, and magnetometer. Which will easily recognize the hand gesture, wrist and finger movement and these 3 sensors will make a unit known as an inertial measurement unit(IMU) and the next unit will compromise of EMG electrodes which will read the electric signals from the IMU unit and then transmit those to nearby bluetooth connected device, as you know that electrical pulse contains noise in it the IMU unit will have an amplifier in it which will remove the unwanted noise. To make the system user independent fuzzy logic would be implemented (a computing technique based on ”degrees of truth”) which will be used to improve the accuracy of the system. The system is still in the development stage.

Similarly, Ankit Ojha *et al* [10] worked on creating a desktop-based application with the help of a webcam that will allow the computer to capture user gestures and then translate them into text and then will further be converted into audio. They mainly focused on image acquisition through the camera which further processed as grayscale images with dimensions of 50*50. Then moving on to the next step the images are scanned for the hand gestures which ultimately became a preprocessing part for better accuracy. Next, the images are fed to the CNN model (deep learning method) which helps to predict the gestures with the highest probability. Then, at last, the model combines those recognized gestures into words, and these words are converted into audio using the pyttsx3 library(text-to-speech library in Python). The CNN model plays a crucial role in image processing. For this project, the CNN model consists of 3 convolution layers 1st layer is used to accept 50*50 images in grayscale, 2nd layer is responsible for the identification of curves and shapes and the last layer is used for gesture identification. The project has an accuracy of 95%.

Similarly, Kanchan Dabre and Surekha Dholay [12] proposed a paper that deals with the implementation of the ML model for interpreting Sign language. The interpretation system works in two phases. In the first phase preprocessing is done using image processing. Here, the hand shape and other features are retrieved from the image utilizing background removal, blob analysis, filtering, noise removal, conversion to grayscale, etc. In the second stage, classification is done into several different gestures using Haar Cascade Classifier, which is trained using samples taken from different angles, positive samples, and negative samples. 500 positive, 500 negatives, and 50 test samples for each gesture. The expected result of this system is fully segmented words or sentences in the text as well as audio format. The Sign Language Interpretation will predict the Indian Sign Language. The average accuracy rate is 92.68%.

likewise, Smit Patel *et al* [13] proposed a work that deals with sign language detection using deep learning techniques. For this project, Microsoft Kinect is used to get depth information along with color images. There is no need for segmenting hands from the background because of using CNN as it can learn features from raw data. The stages involved while approaching Sign Language Recognition consist of choosing the regions of interest, extracting features describing these regions, and with these features training a classifier. CNN fulfills the 3 stages with just a single network trained on a raw pixel-to-classifier output. The model proposed in this paper performs 3D convolutions to learn about the spatial features and the temporal features. The dataset consists of 87000 images, 3000 for each of its 29 classes. The dataset contains all alphabets, ‘space’, ‘nothing’, and ‘delete’ signs. The algorithm used is CNN along with Pooling, and Softmax. Tools used are TensorFlow, Keras, OpenCV, Tkinter, Python Imaging Library, and Numpy. The model is trained using Transferred learning to reduce training time. The pretrained model used is VGG16. The model was tested on 500 new unknown images per character. The final accuracy of around 95.33% is achieved.

On the other hand, Wadhawan *et al* [14] proposed a research paper that deals with robust modeling of static signs in the context of sign language recognition using deep learning-based convolutional neural networks (CNN). The dataset comprises 35,000 images which include 350 images for each of the static signs. There are 100 distinct sign classes that include 23 alphabets of English, 0–10 digits, and 67 commonly used words (e.g., bowl, water, stand, hand, fever, etc.).The model training is based on convolutional neural networks. The proposed model is trained using the Tesla K80 Graphical Processing Unit (GPU), 12 GB memory,64 GB Random Access Memory (RAM), and 100 GB Solid State Drive (SSD). the highest training accuracy of 99.72% and 99.90% on colored and grayscale images.(the training of deep networks occurs in a layer-wise manner and depends on more distributed features as present in the human visual cortex). The steps used in this project were data acquisition, data preprocessing, model training, and testing. The results of the proposed CNN-based sign language recognition system are best when experimentation was performed with different numbers of layers in CNN architecture.

2.2 Limitation of State of the Art techniques

The real-time accuracy was not up to the standards as per the results of various research. Additionally, many of the researcher have restricted their work to alphabets and numbers and less research is done on Indian Sign Language. The audio conversion was also lacking in their proposed models. The work are limited to the computer local host the application built by them are not accessible to the users.

2.3 Discussion and future direction

Future direction includes overcoming of drawbacks of previous work. Most of the research papers were based on converting alphabets and numbers from sign language into text or audio. It was also observed that most of the projects were in a development phase. Methods mentioned in few of the papers did not results into good accuracy. Many of the researchers preferred American Sign Language over Indian Sign Language.So, the proposed work mainly focuses in Indian Sign Language(ISL) the dataset not only consist of alphabets and number but along with it words and phrases is also been included. The system not only converts the recognized gesture into text but also converts it into audio format.

2.4 Concluding Remarks

As per the above literature study, has been observed that different classification techniques / algorithm were used to get more accurate classification / recognition model. To move further, need to work on ISL and need to create own dataset for model training according to improve accuracy. Further Work is extended for gestures of numbers and phrases. Designed system is able to recognize static and dynamic hand gestures.

Chapter 3

Problem Definition and Scope

3.1 Problem statement

To deploy a system to convert Indian sign language into audio to help hearing impaired/deaf people while communicating with the rest of the world

3.2 Goals and Objectives

3.2.1 1st Semester

1. To acquire a deep knowledge of Indian Sign Language.
2. To prepare a data-set of alphabets and numbers in Indian sign language.
3. To select appropriate techniques/algorithms for the detection of hand gestures as well as classify those gestures.
4. To train and test a model on a prepared dataset.

3.2.2 2nd Semester

1. To develop a dataset and model for specific words/phrases in the Indian Sign language.
2. To form text using dedicated alphanumeric characters/words with the help of a designed algorithm.
3. To convert the formed text to audio format.
4. To test the designed model.

3.2.3 3rd Semester

1. To deploy the model over web-app.
2. To test the designed platform.
3. To create draft of research paper.
4. To publish a research paper in Scopus Indexed Conference or Journal.

Chapter 4

System Requirement Specification

4.1 Specific Requirements

The features, functions, and tasks that must be performed for a project to be accomplished are referred to as project requirements. They specify the numerous goals for stakeholders to fulfil and provide everyone involved with a clear set of parameters to work toward.

4.1.1 User Requirements

User needs are exactly same as the name suggests. They are user-defined requirements. These specifications specify how a facility, piece of equipment, or process should perform in terms of the product to be manufactured, needed throughput, and manufacturing conditions. User requirements give information that can be used to further specify, design, and test a manufacturing system. Commissioning and Qualification activities should be designed in such a way that at the end of the process, there is written proof demonstrating that the user criteria have been met.

User requires an dedicated interpreter for Indian Sign Language which recognize hand gestures in Indian Sign Language. Also, text to audio converter allows users to simultaneously get the audio output for the corresponding gesture, this feature ease the efforts of reading text. We have also created GUI which makes the interface user-friendly. This work should be deployed and and accessible anywhere.

4.1.2 Functional Requirements

The project need to implement on system having minimum 32 GB ram, CPU of AMD Ryzen 7 5800H 4GHz and above versions, and GPU of Nvidia GeForce RTX 3070 Mobile. the above specification gives better performance while training the model.

The software requirements includes Jupyter Notebook or similar platform to code the algorithm. A pre-processed dataset is created for model training using OpenCV. The OpenCV supports noise reduction, gradient calculation, non-maximum suppression, and edge tracking by hysteresis in preprocessing.

Chapter 5

Methodology

5.1 System Architecture

The project's fundamental idea is to develop a model that scans the hand gesture using a mobile phone's/ laptop camera and then recognises the gesture. It then further categorize the recognised gestures. Classes denote letters from the number that the model has identified. It will then be transformed into audio using the specified letters. Fig 5.1 depicts the aforementioned procedure.

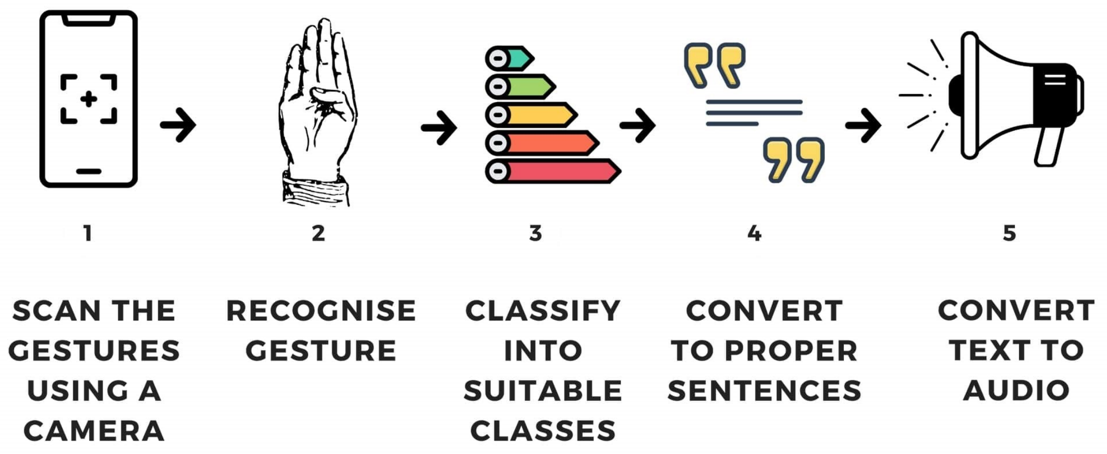

1
SCAN THE GESTURES USING A CAMERA
2
RECOGNISE GESTURE
3
CLASSIFY INTO SUITABLE CLASSES
4
CONVERT TO PROPER SENTENCES
5
CONVERT TEXT TO AUDIO

Figure 5.1: System flowchart

5.2 Mathematical Modeling

Mathematical modelling refers to the mathematical concepts and calculations that done in order to complete a project. Firstly frames from the video feed of camera are extracted and preprocessed. The preprocessed image are stored in a separate directory. For preprocessing Canny Edge is used. Further, for training and classification of the model. CNN based AlexNet is used. For predication of dynamic gestures Long Short Term Memory(LSTM) and Media pipe is used. Using pyttsx3, the recognized text is converted into audio. It is a text-to-speech conversion package that convert recognised gesture text to audio. The block diagram of proposed AlexNet model is shown in Fig.5.2.

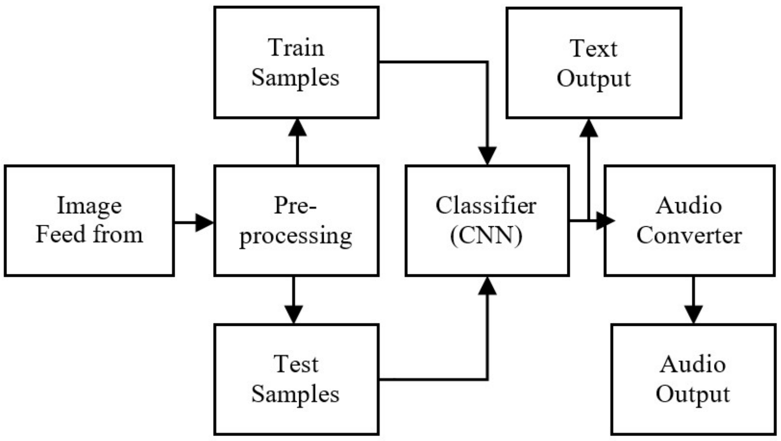


Figure 5.2: Block Diagram of Proposed AlexNet Model

The Mathematical Modeling approach of every stage of the project are explained in the next subsections.

5.2.1 Preprocessing

The quality of the dataset is one of the important factors that affect the accuracy of Deep Learning models. To create datasets and perform preprocessing operations on them,“OpenCV” is used. Various steps involved in preprocessing are , blob analysis background subtraction, noise reduction, grayscale conversion, brightness-contrast normalization, and image scaling. The dataset is created by capturing various different gesture images using webcam of laptop. From the video feed of laptop which is captured at 30 frames per second frames were taken and preprocessing was applied on them and the dataset for specific gesture was stored. The captured frame is in raw form that needs to be cleaned by preprocessing. For preprocessing, the Canny Edge Filtering technique is used[x 15 from research paper]. It consists of five main steps[refer research paper].

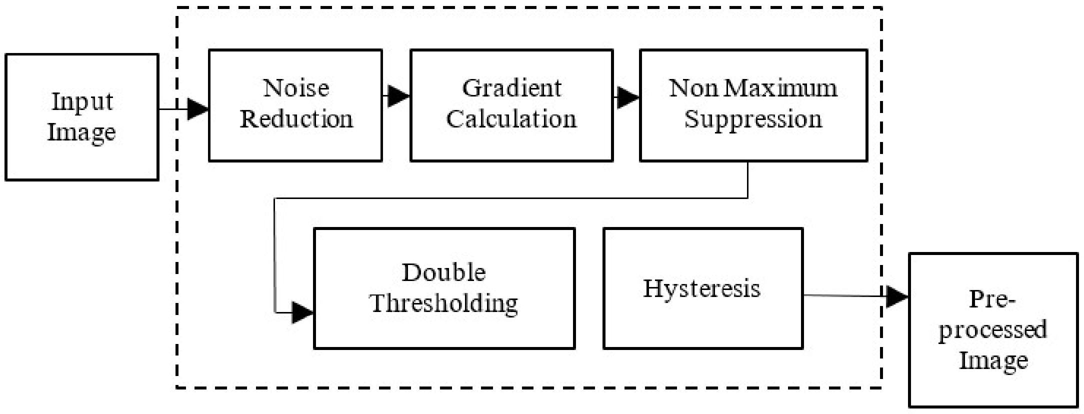


Figure 5.3: Block Diagram for Preprocessing Block

The initial phase is noise reduction by applying a Gaussian blur to smooth the image, In Gaussian blur image is convolved with a Gaussian kernel with the size 3×3 through which a blurred image is obtained. The next step is a gradient calculation which is done using equation 5.1, equation 5.2 and the Gx and Gy are the derivatives obtained from the Sobel kernel :

$$EdgeGradient(G) = \sqrt{G_x^2 + G_y^2} \tag{5.1}$$

$$Angle(\theta) = \tan^{-1}(G_x/G_y) \tag{5.2}$$

A vital step in canny edge detection, non-maximum suppression allows for the output image to have thin edges. This stage produces pixels with the highest gradient values along the edge directions. The produced image has light edges,hence double thresholding is used to create brighter edges. As the output of non-maximum suppression contains few false edges which is not perfect. To overcome this drawback double thresholding is used. In this step two sets i.e. high and low threshold is used. Suppose if high threshold to be 0.8 it states that all the values greater than 0.8 are considered as strong edge. Similarly for low threshold of 0.4 this states that if any pixel is less than this value is not an edge. To determine which weak edges are actual edges hysteresis is used.

Hysteresis is applied towards the end in order to get a clean using edge tracking algorithm. The block diagram of Preprocessing Block is given in Fig. 5.3. With the help of the "OS" module, the preprocessed dataset is stored in the directories named after the corresponding gesture.

5.2.2 CNN (Convolutional Neural Network)

Artificial Neural Networks (ANN) are quite effective among Machine Learning models. ANN are used to classify pictures, sounds, and speech, among other things. Different forms of Neural Networks are used for various reasons, such as predicting word sequences. Recurrent Neural Networks employ a more accurate LSTM, while Convolutional Neural Networks were used for picture categorization [18].

Convolutional Neural Networks (CNNs) are neural networks which exchange parameters. Assume you have a photograph. The concept of an image cube is depicted in Fig. 5.4. It may be expressed as a rectangular box with dimensions of length, width (picture size), and height.

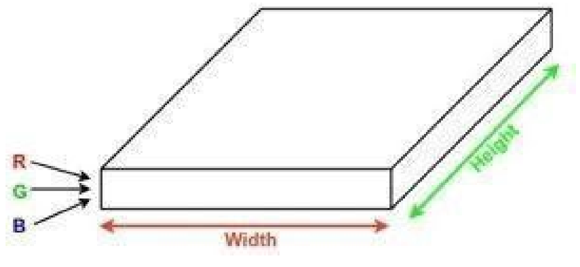


Figure 5.4: Image cube

Consider taking a tiny section of this image and running a small neural network with k outputs on it, then displaying the results vertically. As demonstrated in Fig. 5.5, move the neural network over the picture to generate a new image with varied width, height, and depth. There are now more channels than simply R, G, and B, although they are thinner and higher in height. Convolution is the name given to this procedure. If the patch size is the same as the picture size, the neural network is a normal neural network. There are less weights as a result of this little patch.

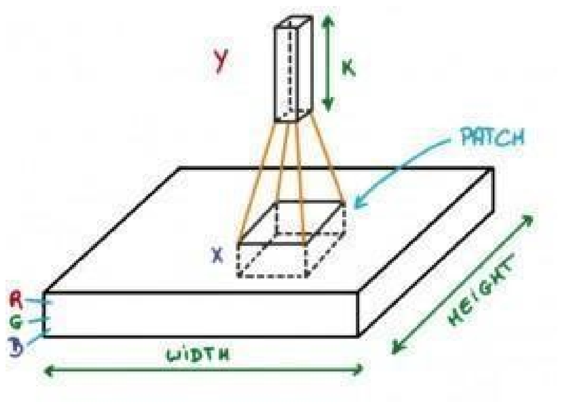


Figure 5.5: Small Patch of Image

Padding essentially increases the region of an image that a convolutional neural network analyses. The kernel/filter that travels over the image searches every pixel and compresses the image, equation 5.3 shows the formula to calculate padding.

$$Padding = inputsize + 2 * padding\ size - (filter\ size - 1) \tag{5.3}$$

CNN layers are composed of a set of learnable filters. Every filter has the same width and height as the input volume, as well as the same depth (3 if the input layer is an image). Assume it wants to execute convolution on a 34 x 34 x 3 image. Filters can be 3x3, where 'a' can also be 3, 5, 7, etc., but they should also be modest in comparison to the picture dimension. During the forward pass, just move each filter across the whole input volume step by step, where each step is referred to as a stride, and then determine the dot product of the filter weights and patch based on the input volume. This will give a 2-D output for each filter after sliding it, which we'll put together to generate an output volume with a depth equal to the number of filters. Each of the filters are learned by the model.

Kernel convolution is employed in many different Computer Vision algorithms, not simply CNNs. It's a procedure in which we select a tiny matrix of numbers (referred to as the kernel or filter), send it across our picture, and then modify it based on the values of the filter. Subsequent feature map values are generated using the method shown in the equation 5.4, where f represents the input image and h represents kernel. The indices of the rows and columns of the result matrix are denoted by m and n, respectively.

$$G[m,n] = (f,h)[m,n] \sum j \sum kh[j,k] f[m-j,n-k] \tag{5.4}$$

After applying our filter to a given pixel, multiply each kernel value in pairs with comparable image values. Finally, add everything up and insert the total in the proper spot on the output feature map.

Convolution over the 6*6 picture using a 3*3 kernel results in a 4*4 feature map. Which is due to the fact that there are only 16 distinct spots in this image that may set our filter. Because our image diminishes with each convolution, we just need to do it so many times before it totally vanishes. It is observed that the influence of pixels on the periphery is considerably lower than those in the center.

To solve both of these problems, just add a frame to the image. For example, if anyone wants to use 1px padding, our photo will be 8x8, as well as the output for the convolution operation with the 3*3 filter will be 6*6. Practically, it is required to fill in the extra padding with zeros. Depending on the way of applying padding or not, two different convolutions are obtained: same and valid. Here naming is pretty awful, therefore for the purpose of clarity: Same means to utilize the boundary around it, so that the size of image at the input and output is same and valid implies the use of original picture. For the first scenario, the width of padding should satisfy the following equation 5.5.

After applying our filter to a given pixel, multiply each kernel value in pairs with comparable image values. Finally, add everything up and insert the total in the proper spot on the output feature map.

$$P = \frac{f-1}{2} \tag{5.5}$$

Step length can be considered another of the convolution layer hyper - parameters. When developing our CNN architecture, steps can be raised so as to make receptive fields overlap less likely, otherwise if someone needs lower spatial dimensions of the feature map, it will be done in a similar fashion. The size of output tensor taking the padding and the stride into consideration - may be determined using the following equation 5.6.

$$n_{out} = \frac{n_{in} + 2p - f}{s} + 1 \tag{5.6}$$

Convolution over volume is a crucial notion that allows us to not only work with RGB or colour images but also to apply multiple filters inside a single layer. The first and most crucial requirement is that the picture to which it is applied and the filter must have the same number of channels. If it is necessary to utilize different filters on the same image, use the convolution for each one independently, stack the results over the another, and merge them together. In the equation 5.7, n is image size, nc is a number of channels in the image, p is used padding, s is used stride, f is filter size, nf is a number of filters.

$$[n, n, n_c] * [f, f, n_c] = \left[\left[\frac{n + 2p - f}{s} + 1 \right], \left[\frac{n + 2p - f}{s} + 1 \right], n_f \right] \tag{5.7}$$

Anyone who has ever attempted to design their own neural network from the ground up understand that the forward propagation accounts for less than 50 percent of the success. When you want to go back, the real fun begins. Nowadays, no need to worry about back-propagation because deep learning frameworks do it. Our objective, with dense neural networks, is to compute derivatives and then utilize these to change the the parameters in the gradient descent.

The chain rule is employed in calculations to determine the influence of parameter changes on the features map and the final output. The complete notation of the partial derivative is used in equation 5.8 instead of the abbreviated one shown below for the partial derivative of the cost function.

$$dA^{[l]} = \frac{\delta L}{\delta A^{[l]}} \; dZ^{[l]} = \frac{\delta L}{\delta Z^{[l]}} \; dW^{[l]} = \frac{\delta L}{\delta W^{[l]}} \; db^{[l]} = \frac{\delta L}{\delta b^{[l]}} \tag{5.8}$$

Here, the goal is to calculate db[l] and dW[l] derivatives linked with current layer parameters - and the value of dA[l -1] which is passed to the previous layer. The first task is to compute the intermediary value dZ[l] as shown in equation 5.9, by computing the derivative of the activation function and applying it on the input tensor. Outcome of this operation will be used later, according to chain rule

$$dZ^{[l]} = dA^{[l]} \ * g'(Z^{[l]}) \tag{5.9}$$

To manage the backward propagation of the convolution, a matrix operation known as full convolution is utilised. It is worth noting that the previously rotated kernel was rotated by 180 degrees during this procedure. This process is specified by equation 5.10, where W represents the filter and dZ[m,n] a scalar corresponding to the partial derivative created by the preceding layers.

$$dA = \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} W.dz[m,n] \tag{5.10}$$

5.2.3 AlexNet

AlexNet is a CNN (Convolutional Neural Network). Alex Krizhevsky was the primary designer of AlexNet. It was published in collaboration with Ilya Sutskever and Geoffrey Hinton, Krizhevsky's dissertation advisor. [18] The model was proposed in a research article titled ”Imagenet Classification with Deep Convolutional Neural Network” in 2012. AlexNet won the Imagenet large-scale visual recognition competition in 2012.

AlexNet has eight layers, each with trainable parameters. The model is composed of 5 layers, beginning from max pooling and continuing with three fully connected layers, each using Relu activation except for the output layer. The layers are depicted in the Fig. -. For the non-linear portion, AlexNet employs ReLu (Rectified Linear Unit) rather than the Tanh or Sigmoid functions that were previously used in typical neural networks.the equation 5.11 describes the ReLu function.

$$f(x) = max(0, x) \tag{5.11}$$

The benefit of ReLu against sigmoid is that it is able to train more quicker since the derivative of sigmoid gets extremely tiny in the saturation region, and hence weight updates essentially disappear as shown in Fig. 5.6. This is referred to as the vanishing gradient problem.

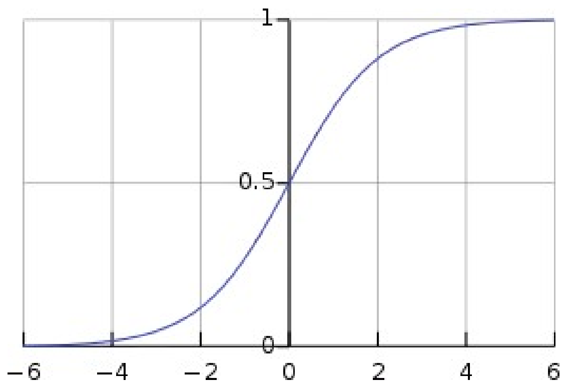


Figure 5.6: Sigmoid Curve

The ReLu layer is placed in the network after each fully-connected layer (FC) and convolutional . Another issue that the AlexNet architecture handled was reducing over-fitting by incorporating a Dropout layer after each FC layer. The dropout layer is coupled with a probability (p), and is applied to each neuron in the response map separately. As shown in Fig. 5.7, it shuts off the activation at random mode with the probability p.

Convolution: The initial convolution layer has 96x11x11 filters with stride 4. The activation function utilised in this layer is relu. The generated feature map is 55x55x96. The output size of a convolution layer is given by equation 5.12:

$$output = ((Input - filtersize)/stride) + 1 \tag{5.12}$$

The channel in the output feature map is determined by the number of filters.

The first Max Pooling layer is of size 3X3 and stride 2. Then the resulting feature map with the size of 27x27x96.The filter size is lowered to 5X5 in the second convolution

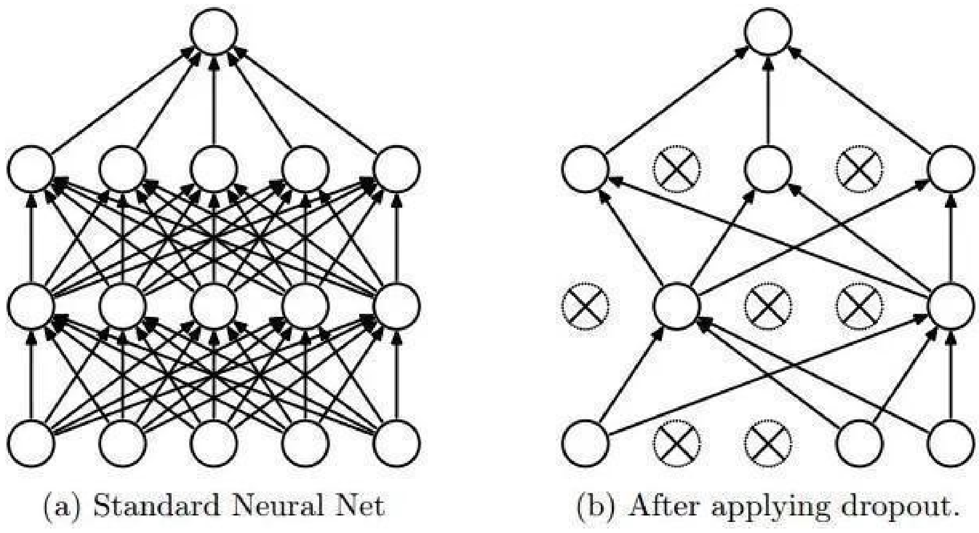


Figure 5.7: Fully Connected layer in CNN with and without dropout layer.

process, and about 256 such filters are available. The stride is one and the cushioning is two. The activation function is ReLu once more. Now the output size is 27x27x256. Again apply a 3X3 max-pooling layer having stride 2 The resultant feature map has the dimensions 13x13x256.

Use 384 filters of size 3X3 stride 1 and padding 1 for the third convolution operation. Activation function is utilised once again. The resulting feature map has the dimensions 13x13x384.

The fourth convolution process uses 384 3X3 filters. The stride with padding is one. Furthermore, the activation function employed is relu. The output size is now same as before which is 13X13X384.

Following that, the final convolution layer of size 3X3 with 256 such filters is generated. Stride and padding are both set to one, and the activation function is set to relu. The resultant feature map has the dimensions 13X13X256.

Following that, the next max-pooling layer of size 3x3 and stride two are given. As an outcome, the 6x6x256 feature map is constructed. The first dropout layer is created here. The drop-out rate is fixed at 0.5 percent. Then comes the formation of the first completely linked layer with a relu activation function. The outcome is 4096 bytes long. Then comes a dropout layer with a fixed dropout rate of 0.5. Following that is a second entirely connected layer with 4096 neurons with relu activation.

Finally, the last fully connected layer or output layer with 1000 neurons is constructed as there are around 10000 classes in the data set. The activation function used at this layer is Softmax. The architecture of the Alexnet mode has a total of 62.3 million learnable parameters [17].

5.3 MediaPipe

Google MediaPipe is an open-source platform for developing word-class machine learning solutions. Currently under alpha testing. It has been open-sourced for a year, although it has most likely been in development for much longer. MediaPipe's code is written in C++, but it can be readily deployed to any platform, from web assembly to Android to Mac OS [20]. MediaPipe's speed is possible due to the utilization of GPU acceleration and multi-threading. Such development procedures are normally challenging, but MediaPipe takes over and accomplishes them for you as long as you follow excellent graph-making principles. The multi-threading and GPU acceleration allow newer phones to run away with frames, often being at FPS too high to see with the human eye.

Because MediaPipe employs graphs, subgraphs, and calculators. The work of one project may simply be transferred to the work of another. When combined with side pockets, anyone may genuinely customize the specifications of each calculator to meet different tasks.

MediaPipe currently has a wealth of ”sample calculators” that you may freely use, such as multi-platform renderers, multi-platform TensorFlow Lite, and pre-built neural networks [20].

Firstly, video feed from camera is taken which is operated at 30 fps (frames per second) then the video frames is taken then key points extraction using Media-pipe is done then key-points are stored in NumPy file then further they are divided into train and test samples and given to LSTM network for classification. The output of LSTM network is text in form and it is also converted into text using a library name pyttsx3. The Block diagram of proposed Media Pipe model is shown in Fig. 5.9.

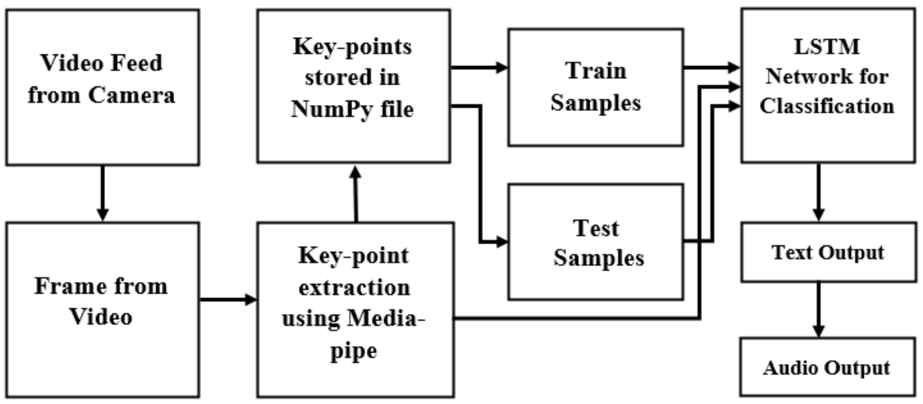

Video Feed from Camera
Frame from Video
Key-point extraction using Media-pipe
Key-points stored in NumPy file
Train Samples
Test Samples
LSTM Network for Classification
Text Output
Audio Output

Figure 5.8: Block Diagram of proposed Media pipe model

Multi-platform support, including Windows, Mac, and Ubuntu, can be a daunting endeavour for a tiny development team. However, within few hours, you may deploy your application not only to desktops but also to mobile devices.

5.3.1 LONG SHORT TERM MEMORY (LSTM) NETWORK

Long Short-Term Memory (LSTM) networks are type of recurrent neural network (RNN) which are used in sequence prediction problem statement. Recurrent units in LSTM are significantly more complicated than those in RNN, which enhances learning but uses up more computational power. It is used to reduce or avoid long-term dependency problem.

LSTM model has chain like structure which can be seen in Fig X.X and four layers of LSTM interact in a special way. Cell state is the key to the LSTM and it is like conveyor belt as it runs linearly with some linear interactions. Also the information flows along it with no change. Also the network has to feature to add or remove information to cell state which are regulated by gates.

For every step RNN loops back the state of that hidden layer, so that the next time step has the memory of last step. If a sequence of length 21 as our entry. For the next step, RNN is able to remember previous state. For later state length of sequence is every time a new input is combined with a loop return the state. The idea that they do not have the same strength. Almost the first step resolve after the new input is

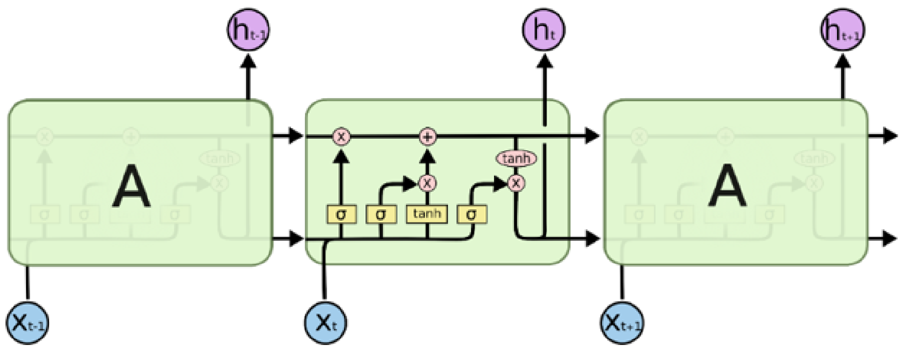


Figure 5.9: LSTM Network

repeated this long. As a result, RNN has a better recall or a better perception of the latest past Short-term memory. The main goal of the LSTM cell was to provide long-term memory to these RNN systems.

5.3.2 PERFORMANCE PARAMETERS

To assess the performance of working models Performance metrics plays very important role. Performance measuring occurs in almost every domain, but it may be accomplished in a variety of ways. Performance parameters are used to measure performance.In this proposed work, the performance parameters used are Confusion Matrix from which accuracy, error rate, precision and recall is calculated.

Confusion Matrix: Using this matrix, it is possible to assess how well categorization models perform given a certain set of test data. It is only possible to identify if the true values of a testing data is known. The matrix itself is straightforward to understand, but the language connected with it may be confusing. Because it displays mistakes in model performance as a matrix, it is also known as an error matrix.

The basic framework of the confusion matrix is shown in Fig. 2,13. As True Positive occurs when the model predicted yes and the actual value was likewise true. False Negative occurs when the model predicted no but the actual value was Yes; this is also known as a Type-II mistake. True Negative occurs when the model predicts No and the actual or true value is likewise No. A false positive occurs when the model

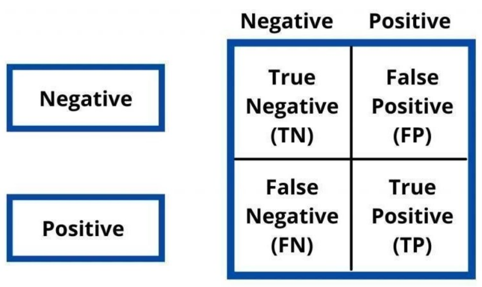


Figure 5.10: Confusion Matrix

predicted Yes but the actual result was No. It's also known as a Type-I mistake. Using this matrix, several operations is conducted for model computations, such as the model's correctness. These computations are provided further:

Classification Accuracy: It is a crucial metric in determining the accuracy of classification tasks. The formula for accuracy is provided in equation 5.13, where the numerator addition of true positive and true negative is considered and the denominator addition of all for values is done. It indicates how regularly the model correctly predicts the output. It could be calculated as the ratio of the classifier's correct predictions to the overall predictions made by the classifiers.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{5.13}$$

Error rate: It is also known as the error rate, because it describes how frequently the model makes incorrect predictions. The formula for error rate is provided in equation 5.14 where the numerator includes the addition of false positive and false negative values and the denominator includes all for values. The error rate is derived by dividing the number of inaccurate predictions by the total number of predictions produced by the classifier.

$$ErrorRate = \frac{FP + FN}{TP + TN + FN + FP} \tag{5.14}$$

Precision: It could be defined as the number of correct outputs supplied by the model or how many positive classifications predicted properly by the model were really true. It may be determined using equation 5.15, where the True Positive value is divided by the total of True Positive and False Negative.

$$Precision = \frac{TP}{TP + FN} \tag{5.15}$$

Recall: It is defined as the percentage of positive classes predicted correctly by their model out of a total of all positive classes. The recall rate should be as high as feasible. It may be determined using equation 5.16, where the True Negative value is divided by the sum of the True Positive and False Negative values.

$$Recall = \frac{TN}{TP + FN} \tag{5.16}$$

F-measure: It is difficult to compare two models that have poor accuracy but good recall, or vice versa. F-score is thus employed for this purpose. This score allows us to examine both recall and accuracy at the same time. The F-score is maximized when the recall equals the accuracy. The F-measure may be computed using equation 5.17.

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{5.17}$$

5.4 Text to Audio Conversion

Text-to-speech conversion is the process of generating computer based human voice using text. There are two ways to convert text to audio. Which are the strategies for online text to audio conversion and offline text to audio conversion? The offline text to audio conversion technology is employed in this research. The pyttsx3 package was used for real-time text to audio conversion. The key characteristics of the previous tx3 library were that customers could select from a variety of voices that were loaded on their machine. Controlling the pace of speech and storing the desired audio to an audio file are both options here.

The live detected text in this project is sent to a user-defined function. Using the pyttsx3 library, an object of the default function Object() [native code] was generated in this function. The parameters of a voice are specified, such as the voice in the system the user wishes to use, the pace of the speech, and so on. Finally, the text provided into the function to audio is converted and instantly output in the form of text is created. The pyttsx3 package is adequate to convert text to audio without internet connectivity by using the system's default sounds.

Python's pyttsx3 is a text-to-speech translator lybrary. It can used offline and it is compatible with both Python 2 and above versions. An application calls the pyttsx.init() factory method to obtain a reference to a pyttsx3. Engine instance is a simple tool which converts input text into voice. The pyttsx3 module supports two voices, one female and one male, both given by ”sapi5” for Windows. It supports three TTS engines which are nsss – NSSpeechSynthesizer on Mac OS X, 3sapi5 – SAPI5 on Windows 2 and espeak – eSpeak on all other platforms [20].

Chapter 6

Implementation

6.1 System Implementation

The most difficult element is developing a platform for deaf and dumb people. The scope of Indian Sign Language is enormous. The CNN model and the MediaPipe concept are employed in this research. TensorFlow, NumPy, and pandas libraries for dataset management, sklearn for accuracy prediction, and matplotlib for graph charting are all essential libraries for Convolutional Neural Networks (CNN). The entire project is written in Python, and Jupyter Notebook is used to train and assess the model. The system configuration required for CNN models is extremely high. For training, the model requires 8 GB of graphics support and around 32 GB of RAM. While a MediaPipe system with a standard setup is adequate.

The hand gesture recognition system in MediaPipe requires the os library, NumPy library, time library, sklearn library, and TensorFlow library. The dataset must then be loaded into the model by providing its path. The add function is used to connect pictures of the same class. The ratio of training and testing for proposed model dataset is 95% and 5% respectively. Further it has been analysed for various training-testing ratio as 70:30, 80:20, 90:10 and 95.05

The flow of platform consists of various methods including Image Feeding, Preprocessing and Classification.

Image Feeding is done from three separate sources. These are the live cameras, the testing dataset, and the training dataset. The camera provides a live feed of photos while utilising a model. Images from the collection are used for training purposes. Photos are collected from the same dataset after training for testing reasons, but not the images used for training.

Several modules were utilized in this section, including the "OpenCV" and "os" modules. The camera of laptop is used to capture the input frames of a video and then the frames are used to produce the image, also the os module is used to create a Folder to store the captured images. In Python, os.mkdir() function is used to build the directory.

Prepossessing: Image preprocessing refers to the steps that are taken to prepare images before they are used in model training and inference. Preprocessing involves actions on pictures at the most basic level of abstraction. Pre-processing is a method for refining image data by suppressing undesired distortions or increasing specific visual components that are important for even more processing. The lowest level abstraction occurs during preprocessing. Let's go over the different preparation stages for this project.

This model will use data from the camera's picture, training samples, and test samples. Initially, they will train and test the model using the provided dataset. Images captured by the camera will be preprocessed by their predefined model before use [6]. The initial phase of preprocessing will be noise reduction, followed by background subtraction, and finally grayscale conversion and scaling of those photos. The preprocessed picture acquired at the last level of data preprocessing is shown in Fig. 6.2. Then, using a pre-trained CNN model, they will categories gesture photos into appropriate classifications. It will generate correct words and phrases based on the recognized alphabets. They're going to add text to the audio conversion feature to save the effort of reading the text.

Classification is done using AlexNet Model and LSTM network trained on their respective dataset. For training Python is used on Jupyter Notebooks. For implementing the AlexNet and LSTM network tensorflow library is used. The model is saved in JSON format and the weights are stored in hdf5 format to use for classification.

6.2 Flowcharts

6.2.1 Flowchart of Dataset Creation for AlexNet Model

The first step in producing the dataset is to set up the folders. The application then catches the hand gesture using the camera. Several photos are captured here. So they must move and spin their hands with hand motions. After that, repeat all of the hand movements. After obtaining the appropriate amount of samples or photos, the procedure should be terminated. The flowchart for creating a dataset in Python is shown in Fig. 6.3.

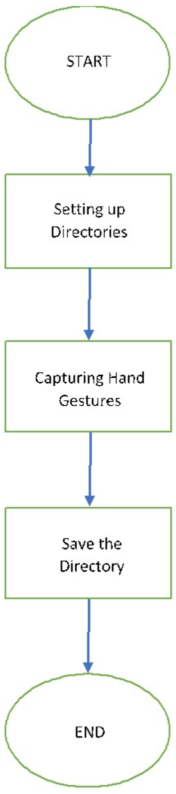


Figure 6.1: Flow Chart of Proposed AlexNet Model

6.2.2 Flowchart For Dataset Creation for MediaPipe Model

To begin with the dataset creation using media-pipe video input is taken from the camera and then from the frames 21 key-points are extracted using media-pipe for each hand.The extracted key-points are stored in the Numpy formats and the whole process is repeated for each dyanamic gestures.The flowchart fro media-pipe based dataset creation is shown in Fig 6.4.

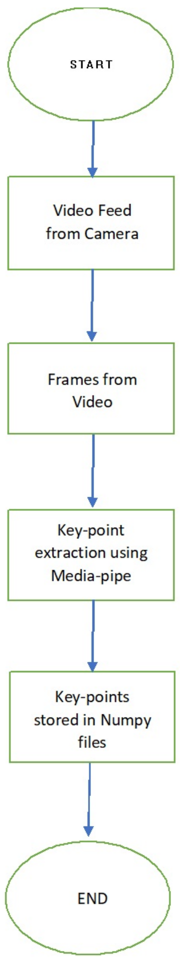

START
Video Feed from Camera
Frames from Video
Key-point extraction using Media-pipe
Key-points stored in Numpy files
END

Figure 6.2: Flowchart For Dataset Creation for MediaPipe Model

6.2.3 Flowchart for Proposed MediaPipe Model

The Numpy format key-point extracted dataset is passed on to the LSTM classifier over which the model is trained and evaluated with the test samples. Also the performance of the model is analysed. The LSTM based model flowchart is shown in Fig 6.5.

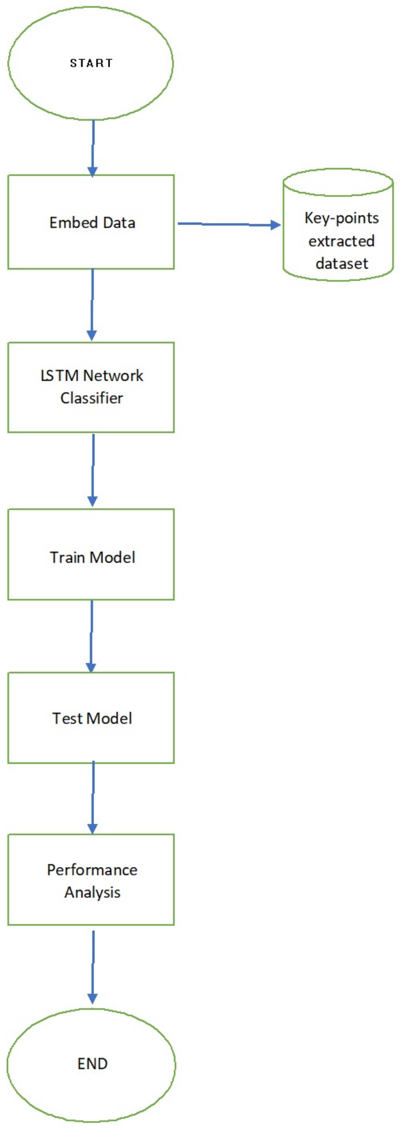


Figure 6.3: Flowchart for Proposed MediaPipe Model

Chapter 7

Result Analysis/Performance Evaluation

The outcomes of the two separate methods utilised for the recognition of sing-language are presented in this section. The results from each and every step for both models are displayed and further discussed with comparative analysis, which explains that the best technique for static gesture recognition is Alexnet, while the best approach for dynamic gesture recognition is Media Pipe.

7.1 Results of AlexNet Method

In Alexnet based approach, the dataset is created by using OpenCV in parallel the image is preprocessed using canny edge detection. The generated dataset is then trained over different training testing ratio of 70:30, 80:20 , 90:10 and 95:05.Further the performance analysis was done using confusion matrix and calculating different performance parameter's. Further the results for each step is discussed

Dataset Creation and Pre-processing

For pre-processing canny edge detection is used. In canny edge processing, firstly the noise reduction and grayscale conversion of the input image (A) takes place and the output image looks as shown in block (B) from Fig 7.1. Once the noise from the input is reduced a Sobel filter is applied to compute the gradients and the output image from this step resembles that in block (C).To obtain the thin images and remove another patch Non-Maximum Suppression is applied and the output image here resembles that in block (D) and at the end to get bright edges double thresholding and hysteresis are done and the final output image resemble that in block (E).

Fig. 7.1. shows the detail steps of image preprocessing done in canny edge. Initially, Captured image for the dataset was RGB format and canny edge detection is applied on it. Canny edge detection involves Noise Reduction, Sobel filtering, Non-Maximum suppression and double thresholding followed by hysteresis.

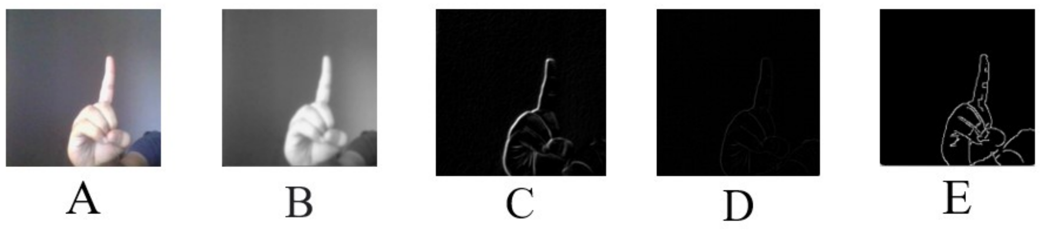


Figure 7.1: Image pre-processing steps (A) Input Image (B) Noise Reduced Reduced Image (C) Sobel Filter Image (D) Non-Maximum Suppression (E) Double thresholding and Hysteresis

.

For this project, three different types of gestures were used. These are Alphabets, numbers and phrases. Figure 7.2. shows detailed preprocessing of all the three types of gestures as samples.

As we are using the AlexNet Model and the input to AlexNet is a black and white image of size 225×225 as shown in Figure 7.3.

A total of 4000 images is stored which consist of 10 classes having 400 images each, that includes alphabets (‘A’, ’B’, ’C’, ’J’, ’Y’), numbers (‘1’,’2’,’3’) and phrases (‘my’,’name_is’) then this dataset is passed down to Convolution neural network

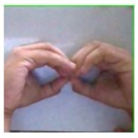
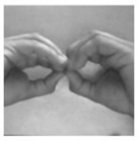
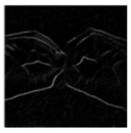
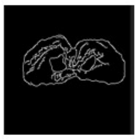
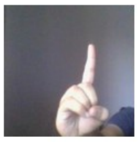
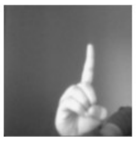
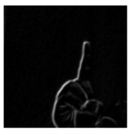
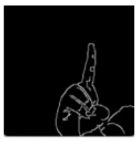
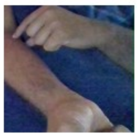
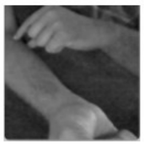
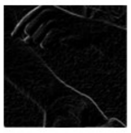
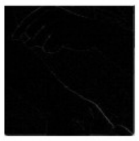
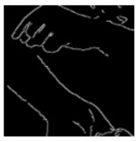
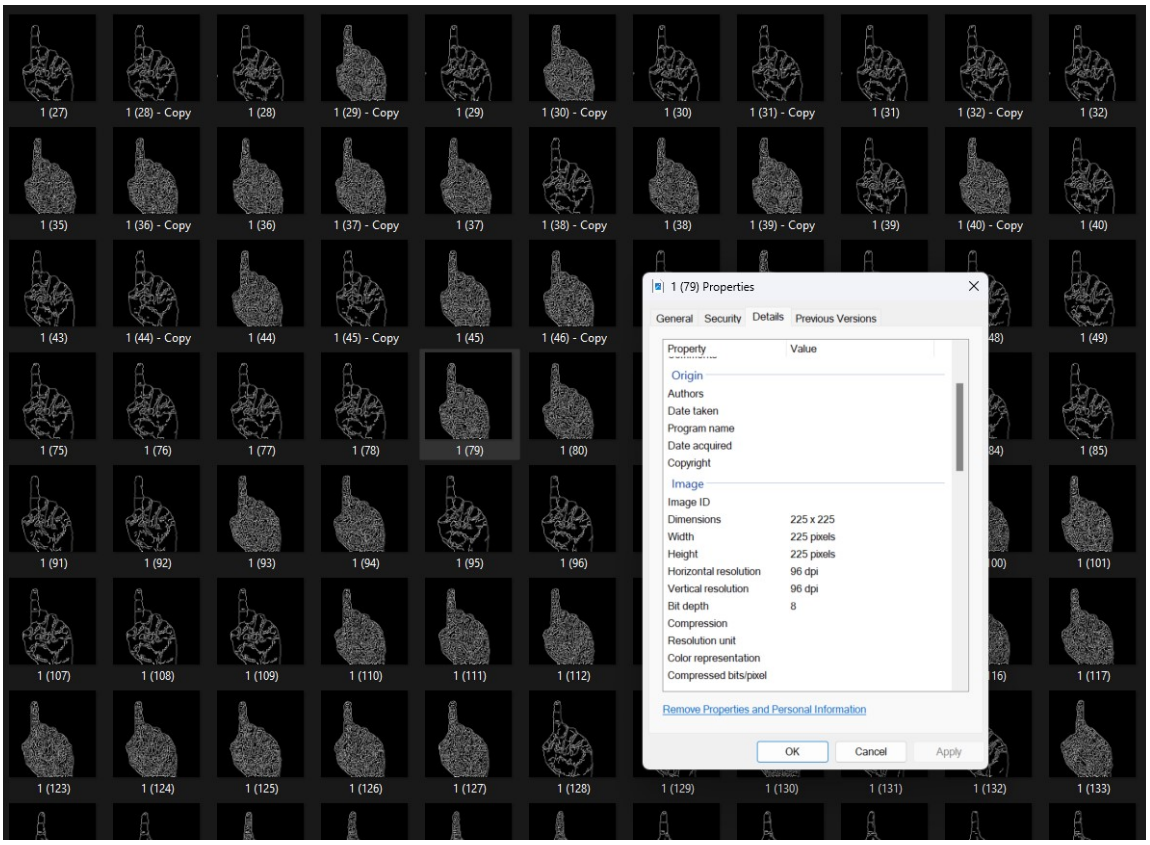
Sample Case	Input Image	Noise Reduced Image	Sobel Filter Image	Non-Maximum Suppression	Double thresholding and Hysteresis
Alphabet (B)					
Number (1)					
Phrase (Name)					

Figure 7.2: Output of various stages of preprocessing



1 (79) Properties

Property	Value
Origin	
Authors	
Date taken	
Program name	
Date acquired	
Copyright	
Image	
Image ID	
Dimensions	225 x 225
Width	225 pixels
Height	225 pixels
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	8
Compression	
Resolution unit	
Color representation	
Compressed bits/pixel	

Figure 7.3: Image Dimension

(CNN) based Alexnet architecture for classification purpose.

7.1.1 Classification

In the case of classification of gestures, the preprocessed image feed is passing to the AlexNet and the corresponding text output is obtained. Figure 7.4. shows the output of real-time prediction and classification. It shows the model output for the corresponding input gesture. It can be seen that the model accurately predicted the sign "B" and that in addition to providing text output, it also converts the classified text output to audio. As observed from the figure the proposed work has a small constraint regarding the background. Similarly, the model is trained and tested for all the signs of alphabets, numbers and words used in the proposed work.

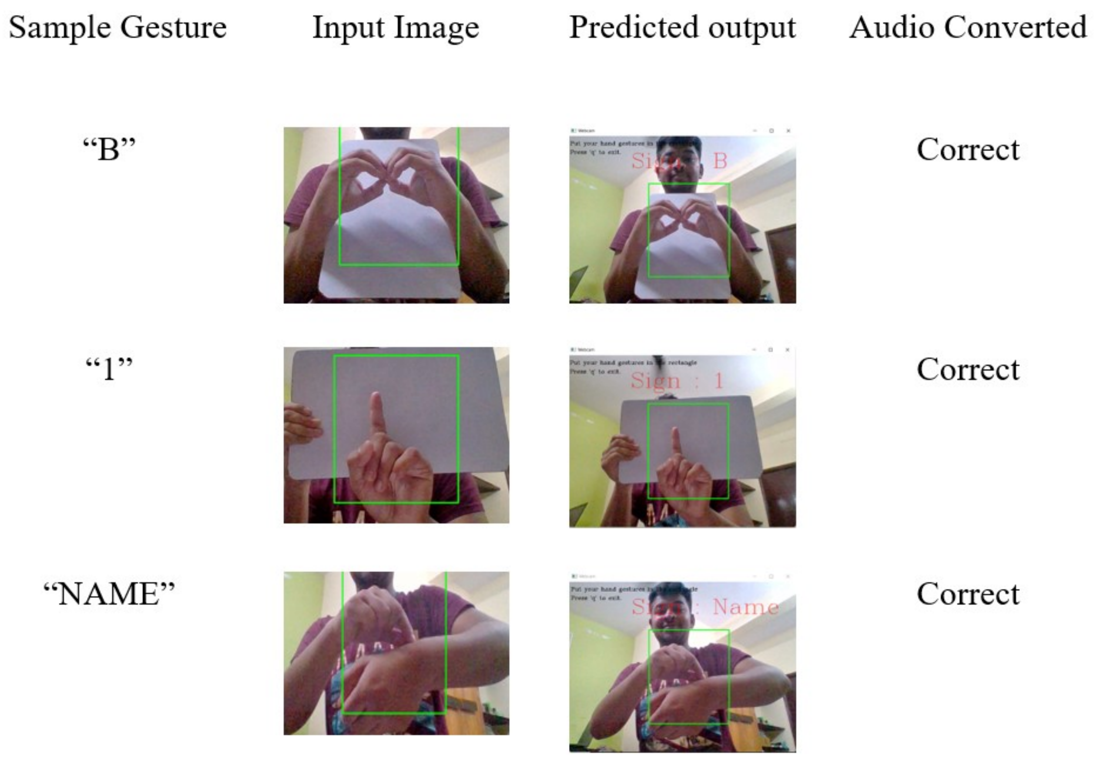


Figure 7.4: Predict Sign after classification

7.1.2 Performance Analysis of AlexNet Based Model

To verify the performance of the proposed model, the performance parameters used are accuracy, recall and precision which can be calculated from the confusion matrices. The model is also analyzed for various training-testing ratios which are 70:30, 80:20, 90:10 and 95:05. After training and testing the model, confusion matrices for all four ratios are shown in Fig.7.5.

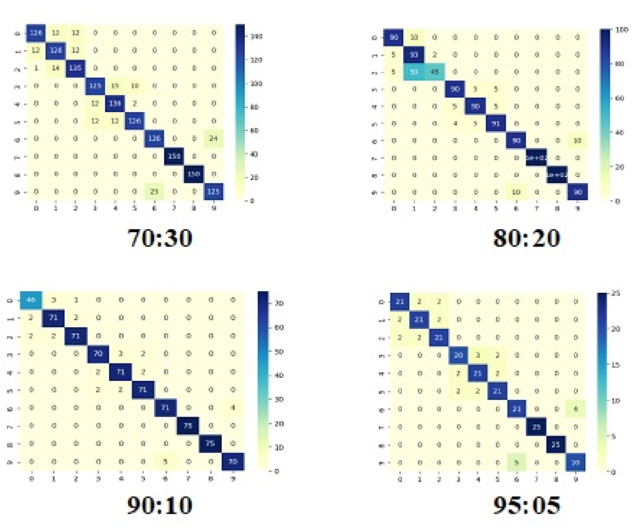


Figure 7.5: Confusion Matrix for 70:30, 80:20, 90:10, 95:05 Training-Testing Dataset Ratio

Based on these confusion matrices, the precision and recall values for all the gestures used in the proposed model are calculated and the results are presented in Table 7.1.

Table 7.1: Accuracy Calculation of AlexNet Model

Classes	**TP**	**TN**	**FP**	**FN**	**Accuracy**
‘1’ : 0	75	0	0	0	100
‘2’ : 1	75	0	0	0	100
‘3’ : 2	75	0	0	0	100
‘a’ : 3	75	0	0	0	100
‘b’ : 4	75	0	0	0	100
‘c’ : 5	75	0	0	0	100
‘j’ : 6	75	0	0	0	100
‘my’ : 7	75	0	0	0	100
‘name_is’ : 8	75	0	0	0	100
‘y’ : 9	75	0	0	0	100

From Table 7.2, it is observed that for the training, and the testing ratio of 90:10 the precision and recall value for all the used gestures is maximum as compared to other ratios. Similarly, the overall model accuracy for different training and testing ratios are calculated and presented in Table 7.1.

Table 7.2: Training and Testing Accuracy for Different Dataset Ratios

Sr. No.	**Training-Testing Ratio**	**Training Accuracy**	**Testing accuracy**
1	70:30	92.11	88.3
2	80:20	94.3	87.9
3	90:10	94.9	95.31
4	95:05	95.47	86.4

Figure 7.5 shows training and testing accuracy for different Training-Testing split ratios of datasets. The different combinations of training and testing datasets are 70:30, 80:20, 90:10 and 95:05. It is observed that as a number of images in the training dataset increases the training accuracy also increases. But, the observed testing accuracy was highest for the 90:10 train-test split ratio.

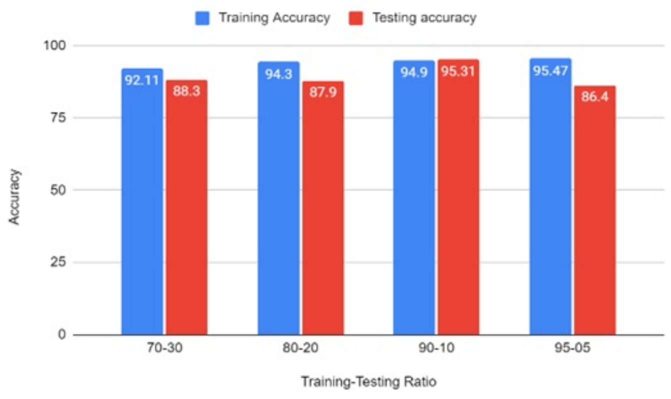


Figure 7.6: Confusion Matrix for 70:30, 80:20, 90:10, 95:05 Training-Testing Dataset Ratio

7.2 Results of Mediapipe - LSTM Method

7.2.1 Dataset Creation

For Mediapipe and LSTM method, the dataset consists of dynamic gestures i.e. videos. The OpenCV library, NumPy library, os library, matplotlib library, and MediaPipe library were used to create a dataset for MediaPipe. The dataset in this case is a NumPy array. To extract key points using Mediapipe, an object is generated for holistic models and drawing tools. Different functions were built for media pipe detection, drawing landmarks of the face, left hands, and right hand, and styling landmarks with colour, thickness, and circle radius parameters. Then, important key points are extracted and labelled in the form of a NumPy array. In Fig 7.6, each step in creation of dataset for the Mediapipe - LSTM method is illustrated for the sign ”temple”.

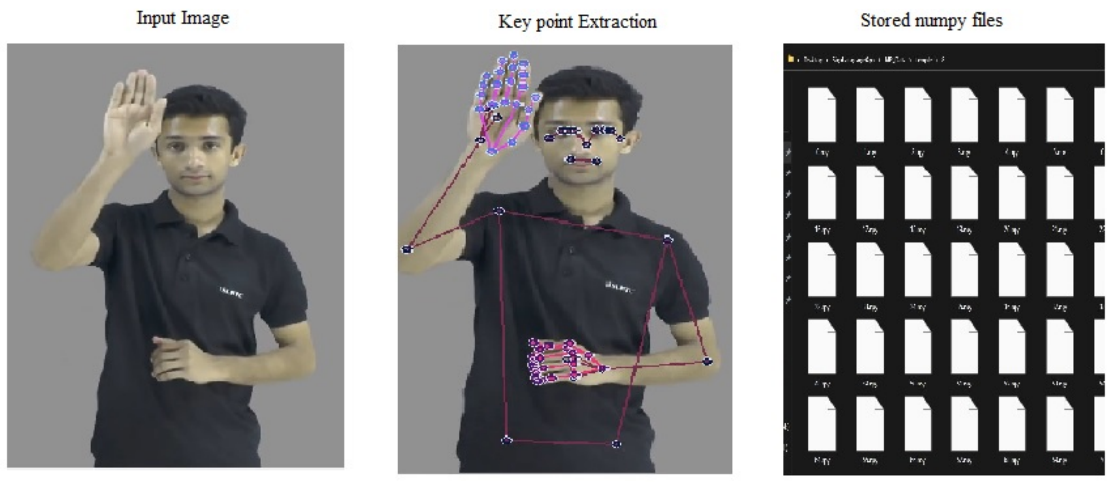


Figure 7.7: Create Sign Interface

The dataset consists of extracted key points stored as numpy arrays, from 30 videos of each class consisting of 75 frames at frame rate of 25 frames per second.

7.2.2 Classification

In (Long Short Term Memory) LSTM, three LSTM layers with Relu activation function and two dense layers with the same activation function are employed for classification. The numpy array of extracted keypoints from frames of video are feed in the LSTM network which gives the predicted sign as an output

7.2.3 Performance Analysis

The model is also analyzed for various training-testing ratios which are 70:30, 80:20, 90:10 and 95:05. After training and testing the model, confusion matrices for all four ratios are shown in Fig.7.5.

Based on these confusion matrices, the precision and recall values for all the gestures used in the proposed model are calculated and the results are presented in Table 7.3.

Table 7.3: Precision and Recall Values of all the signs

		70:30		**80:20**		**90:10**		**95:05**	
Sr. No.	**Sign**	***P***	***R***	***P***	***R***	***P***	***R***	***P***	***R***
1	**'1'**	70	84	90	90	92	92	84	84
2	**'2'**	82.89	84	60.78	93	93.42	94.66	84	84
3	**'3'**	84.9	90	95.74	45	95.94	94.66	84	84
4	**'A'**	83.89	83.33	90.9	90	94.59	93.33	83.33	80
5	**'B'**	83.43	90.66	90	90	93.42	94.66	80.76	84
6	**'C'**	91.3	84	90.09	91	94.66	94.66	84	84
7	**'J'**	83.44	100	90	90	93.42	94.66	80.76	84
8	**'MY'**	100	100	100	100	100	100	100	100
9	**'NAME'**	100	100	100	100	100	100	100	100
10	**'Y'**	83.89	83.33	90	90	94.59	93.33	83.33	80
Note: P – Precision, R - Recall									

From Table 7.3, it is observed that for the training, and the testing ratio of 90-10 for the precision and recall value for of the used gestures is maximum as compared to other ratios. Similarly, the overall model accuracy for different training and testing ratios are calculated and presented in Table 7.3.

7.3 Comparison of two methods in real-time testing

For real time testing, samples of signs were selected. It constituted of different signs for the AlexNet based models. The test was done with varying backgrounds to see the performance of background on the model. The results were checked for both prediction to text and conversion to audio. The results are available in Table 7.3.

Table 7.4: Real-Time test on AlexNet

	AlexNet Model				
Sr. No.	**Actual Sign**	**Background**	**Predicted Sign**	**Converted Audio**	**Result**
1	1	White	1	1	TRUE
2	1	Normal	1	1	TRUE
3	2	White	2	2	TRUE
4	2	Normal	3	3	FALSE
5	3	White	3	3	TRUE
6	3	Normal	3	3	TRUE
7	4	White	4	4	TRUE
8	4	Normal	3	3	FALSE
9	5	White	5	5	TRUE
10	5	Normal	5	5	TRUE
11	My	White	My	My	TRUE
12	My	Normal	My	My	TRUE
13	Name	White	Name	Name	TRUE
14	Name	Normal	Random	Random	FALSE
15	J	White	J	J	TRUE
16	J	Normal	Random	Random	FALSE
17	A	White	A	A	TRUE
18	A	Normal	Random	Random	FALSE
19	Y	White	J	J	FALSE
20	Y	Normal	Random	Random	FALSE

While performing real time testing for the AlexNet model it is observed that the background highly affected the prediction of signs. whilst some signs were predicted perfectly with white clean background, their prediction in normal day to day background produced false results. The complexity of signs also affected their prediction, some complex signs like “Name” and “Y” produced False predictions even for the white clean background.

Similarly, the Mediapipe methods was tested in realtime situation with varying backgrounds to see the performance of background on the model. The results were checked for both prediction to text and conversion to audio. The results are available in Table X.X.

Table 7.5: Mediapipe Realtime Testing Results

	Mediapipe Model				
Sr. No.	**Actual Sign**	**Background**	**Predicted Sign**	**Converted Audio**	**Result**
1	“Take a photo”	White	“Take a photo”	“Take a photo”	TRUE
2	“Take a photo”	Normal	“Take a photo”	“Take a photo”	TRUE
3	“Take a photo”	Recorded Video	“Take a photo”	“Take a photo”	TRUE
4	“Talk”	Normal	“Talk”	“Talk”	TRUE
5	“Talk”	White	“Talk”	“Talk”	TRUE
6	“Talk”	Recorded Video	“Talk”	“Talk”	TRUE
7	“Temple”	White	“Temple”	“Temple”	TRUE
8	“Temple”	Normal	“Temple”	“Temple”	TRUE
9	“Temple”	Recorded Video	“Temple”	“Temple”	TRUE

For the real time tests on the Mediapipe LSTM Model the accuracy came out to be 100%. This result is completely opposite to the tests performed on the dataset in the testing section. This happens due to variations in the background in the real world which are not captured in the dataset.

As per the above results it is observed that Alexnet Model Performs better compared to MediaPipe Model when testing the trained model using the split dataset itself. But in real time scenario it is observed that MediaPipe Model performs better as it is not affected from background variation .

The problem of background is solved in the Mediapipe model. The real time testing of the Mediapipe model was done on dynamic / moving signs. So, the real time testing shows that background did not affect the mediapipe model.

7.4 GUI Implementation

A GUI was implemented for both of the techniques described above for ease of use of the user. The Python library "Tkinter" was used to implement the GUI. We can see the implemented GUI in the Fig. 7.7. The GUI provides several buttons that perform particular tasks. This makes understanding the GUI much easier.

The GUI consist of different features which ease the user to perform operations such as "Create sign", "Predict Sign to Text", "Predict Sign to Audio" and so on.Each and every features are briefed in detailed and the backend operation which are related to each features are given below.

The "Create Sign" button is used to add more signs to the Sign Language dataset.When we press the "Create Sign" button at the backend, the function named Create sign is called and the code for creating sign/gesture is written inside that. The Functioning of this button for creating dataset for AlexNet model can be seen in Fig 7.9.

The "Predict Sign To Text" button Converts the sign detected in the real time video to text.When we press the "Predict Sign To Text" button at the backend, the function named Predict Sign To Text is called and using this function predicted

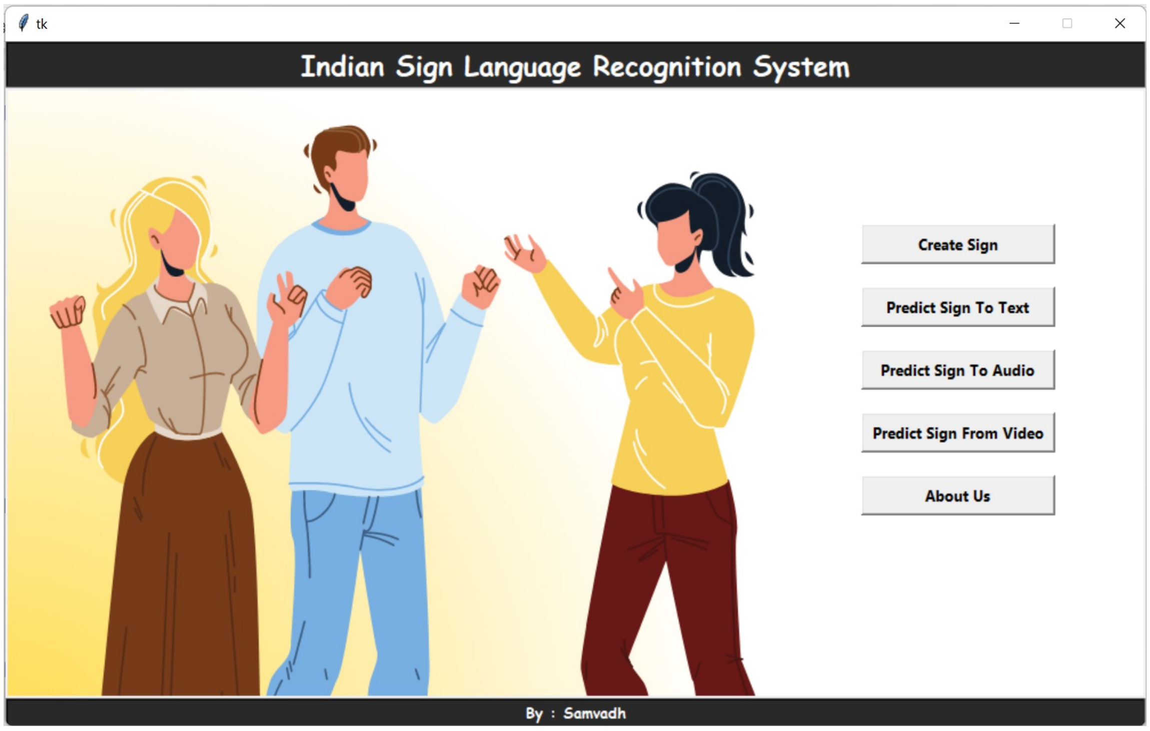
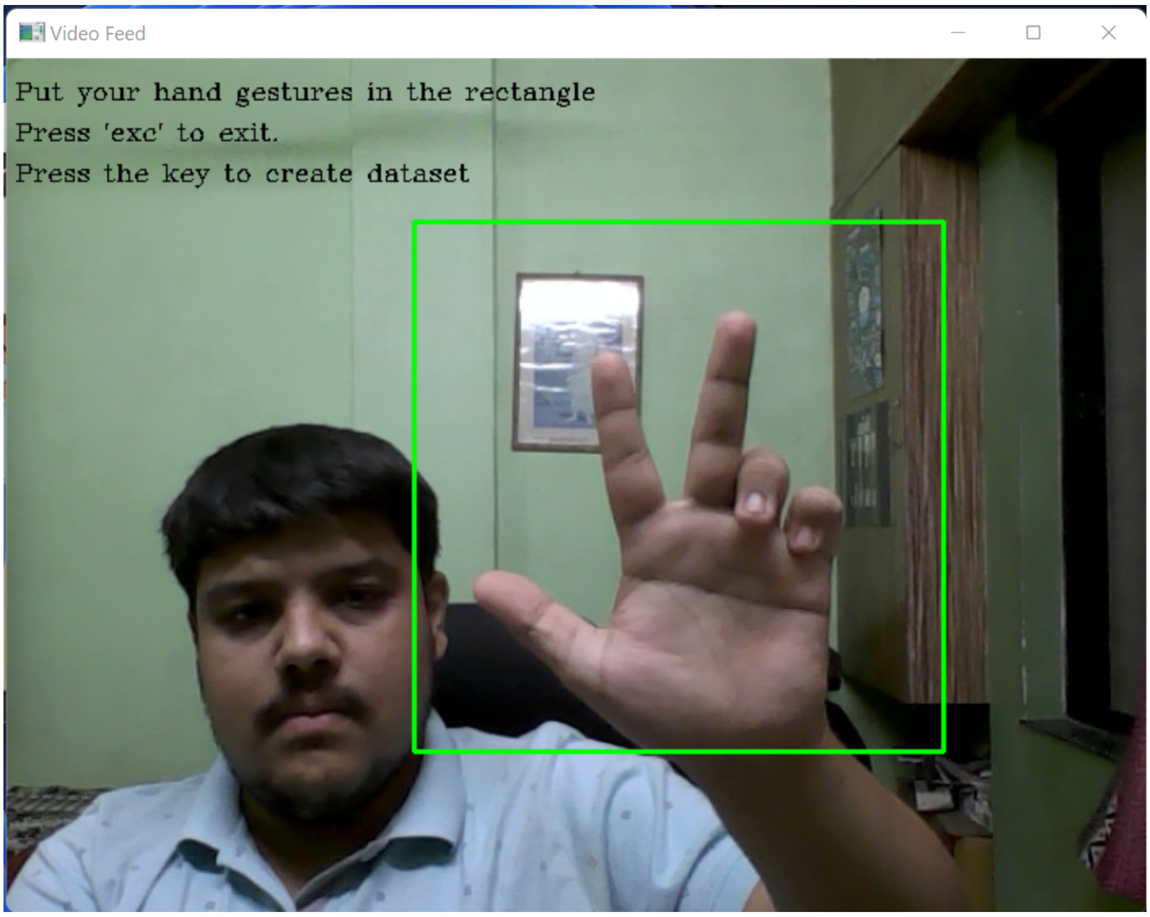


Figure 7.8: GUI Interface



Put your hand gestures in the rectangle
Press 'exc' to exit.
Press the key to create dataset

Figure 7.9: Create Sign Interface

output is converted to text .The Functioning of this button for predicting text using AlexNet can be seen in Fig. 7.10. The “Predict Sign To Audio” button Works similar to the “Predict Sign to Text” button but in addition to text it also outputs the text converted to Audio.

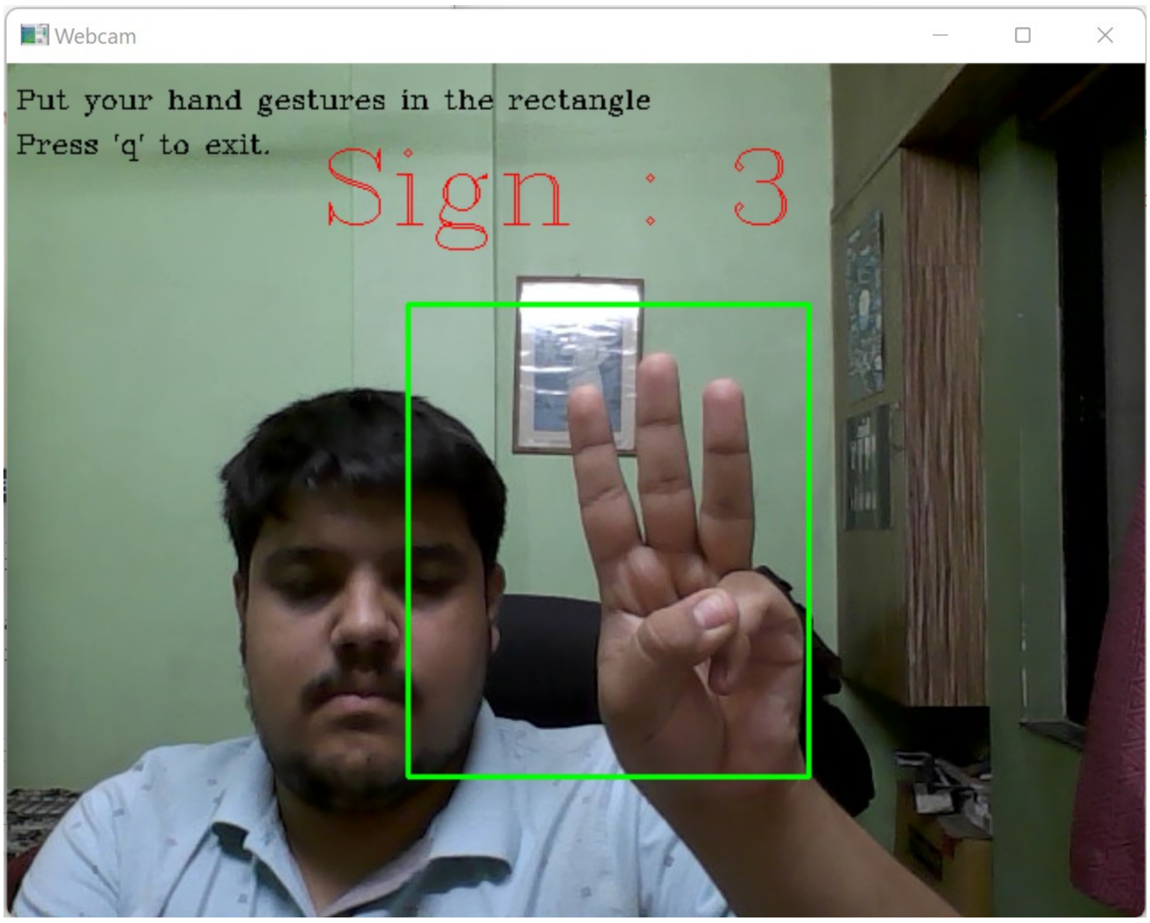


Figure 7.10: Predict Sign Interface

The “Predict Sign From Video” button Converts the sign detected in a prerecorded video to text. When the ”Predict Sign From Video ” button is pressed at the backend, the function named Predict Sign From Video is called and the code for prediction from video is written inside that. The Functioning of this button for predicting text using Mediapipe can be seen in Fig. 7.11.

The ”Predict Sign to Audio” button converts the detected sign into the audio.When the ”Predict Sign to Audio” button is pressed at the backend, the function named Convert text to Audio is called and the function consist of a python library named ”pyttsx3” which converts predicted text to the audio.

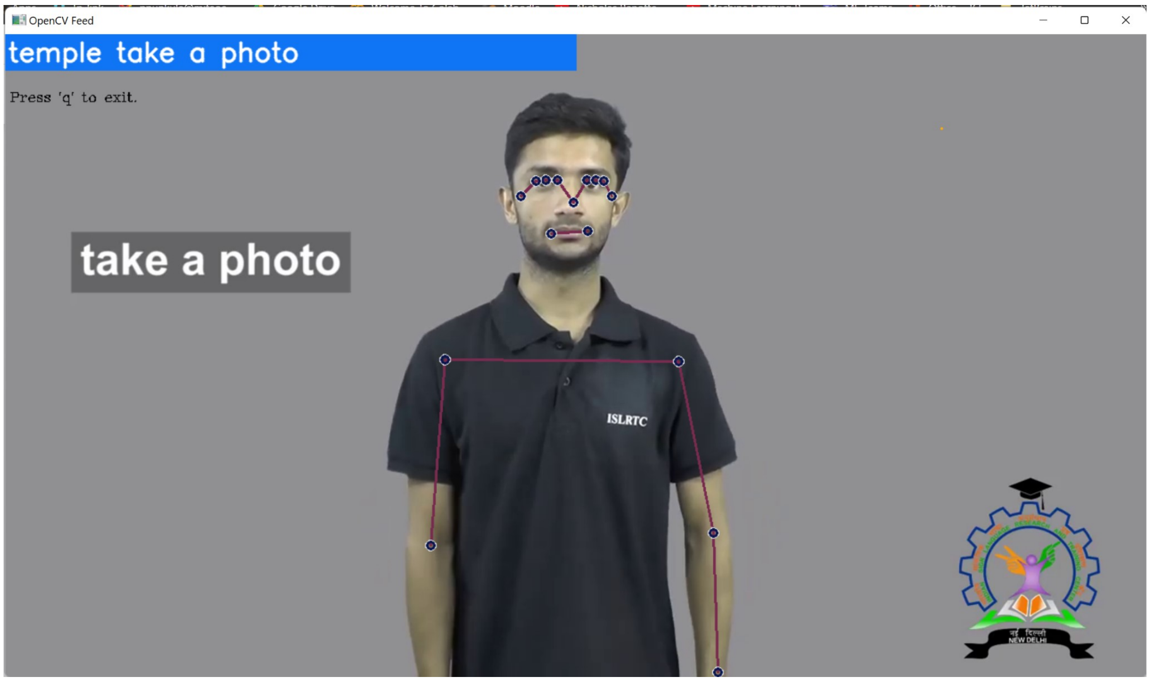


Figure 7.11: Predict Sign From Video Interface

Chapter 8

Conclusion

8.1 Conclusion

For the detection and prediction of signs in the Indian Sign Language we have used a Convolution Neural Network (CNN) i.e. the "AlexNet architecture based CNN " and also the "Meadiapipe based Long Short Term Memory (LSTM) Network". We have created the dataset ourselves consisting of 2000 images of each sign for some signs in the ISL for the training and testing of the "AlexNet based Model". For training and testing of LSTM Network, Numpy arrays of key points were extracted for each class using Mediapipe from 30 videos consisting of 75 frames. Both the "AlexNet based CNN" and "Mediapipe based LSTM Network" were implemented in python using TensorFlow library. Mediapipe and LSTM were used to improve on the drawbacks of the conventional CNN approach.

The overall accuracy of the methods, when tested on the test set, came out to be 98 percent for "AlexNet based CNN" and 81 percent for the "Mediapipe based LSTM Network". This changed when performing real time tests, the "Mediapipe based LSTM Network" performed much better than the "AlexNet based CNN" which was affected by the variations in the background. The "Mediapipe based LSTM Network" achieved accuracy of 100% in the real time tests while the "AlexNet based CNN" was only able to achieve accuracy of 65%. But our current "Mediapipe based LSTM Network" only works on signs with certain duration. We have only trained and tested it for signs with 3 seconds duration or 75 frames in total. Some signs in the Sign Language go on for a longer duration, almost 25 seconds or more, which poses a challenge for including them in our model due to their complexity and storage required for such long signs. Further research and working on Mediapipe and LSTM is required for inclusion of such signs with variable durations and making the project suitable for the majority of the Sign Language.

8.2 Future Scope

Their final product can be used at various Government offices, Public places, also at various public transport to help hearing-impaired people. Also, try to create an app that can be used to solve the communication gap.

Appendices

Appendix A

Appendix

Appendix B

Sponsorship Certificate

Appendix C

Publications/ Achievement Certificate /tPatent

Appendix D

Plagiarism Report of Text

References

Ankit Ojha, S. M. A. T. D. D. P., Ayush Pandey. (2017). Sign language to text and speech translation in real time using convolutional neural network. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, *8*.

Ashish Sharma, S. S., Anmol Mittal, & Awatramani, V. (2020). Hand gesture recognition using image processing and feature extraction techniques. *Procedia Computer Science*, *173*.

Bantupalli, K., & Xie, Y. (2018). American sign language recognition using deep learning and computer vision. *IEEE International Conference on Big Data*, *43*.

Convolutional neural networks. (n.d.). Retrieved 2022-06-15, from `https://www.ibm.com/cloud/learn/convolutional-neural-net`

Dabre, K., & Dholay, S. (2014). Machine learning model for sign language interpretation using webcam images. *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, *7*.

Dutta, K. K., & Bellary, S. A. S. (2017). Machine learning techniques for indian sign language recognition. *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*.

D. Vishal, K. N. B. T. R., H. M. Aishwarya, & Ramesh, T. K. (2017). Sign language to speech conversion. *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*.

Estrada Jiménez L.A., S. N., Benalcázar M.E. (2016). Gesture recognition and machine learning applied to sign language translation. *VII Latin American Congress on Biomedical Engineering CLAIB*.

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow.* O'Reilly Media.

K. Tiku, A. R., J. Maloo, & R, I. (2020). Real-time conversion of sign language to text and speech. *Second International Conference on Inventive Research in Computing Applications (ICIRCA).*

Mediapipe holistic. (n.d.). Retrieved 2022-07-14, from `https://google.github.io/mediapipe/solutions/holistic.html`

Mehreen Hurroo, M. E. (2020). Sign language recognition system using convolutional neural network and computer vision. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*, *9*.

National theater of the deaf. (n.d.). Retrieved 2022-06-13, from `https://www.britannica.com/topic/National-Theatre-of-the-Deaf`

P. R. Sanmitra, V. V. S. S., & Lalithanjana, K. (2021). Machine learning-based real-time sign language detection. *International Journal of Research in Engineering, Science, and Management.*.

A review of google's new mobile-friendly ai framework: Mediapipe. (n.d.). Retrieved 2022-06-18, from `https://medium.com/swlh/a-review-of-googles-new-mobile-friendly-ai-framework-mediapipe-25d62cd482a1`

Sign language. (n.d.). Retrieved 2022-06-05, from `https://en.wikipedia.org/wiki/Sign_language`

Wadhawan A., P., Kumar. (2020). Deep learning-based sign language recognition system for static signs. *Neural Computation Applications*.