<u>**Project Report**</u>

Sign Language Recognition System with Machine Learning

Submitted in complete fulfilment for award of the Degree of

BACHELOR OF TECHNOLOGY

Submitted by:

Akash Murmu (11800119010)
Samrat Biswas (11800119005)

Under the Guidance of

??

DEPERTMENT OF COMPUTER SCIENCE & ENGINEERING

BIRBHUM INSTITUTE OF ENGINEERING & TECHNOLOGY, SURI, BIRBHUM

STUDENT'S DECLARATION

We here by declare that the project entitled "**Sign Language Recognition System with Machine Learning**" submitted by us, in complete fulfilment of the requirement for the award of the degree of the BTech in Computer Science & Engineering is a record of bonafide project work carried out by us under guidance of **(??)** & **(??)**. We farther declare that this project has not been copied, duplicated, or plagiarised from another paper, journal, document or book and   has   not   been   submitted   to   any educational institute or otherwise for the award of any certificate, diploma, degree or recognition.

With Regards

<table>
<tr><td>…………………</td><td>…………………</td></tr>
<tr><td>Akash Murmu</td><td>Samrat Biswas</td></tr>
<tr><td>Roll no - 11800119010</td><td>Roll no - 11800119005</td></tr>
</table>

ACKNOWLEDGEMENT

ABSTRACT

Conversation with a person having a hearing or speech disability has always been a challenge. Sign language has undoubtably become the major tool for the people with hearing and speech disability to communicate their thoughts and feeling to the world. It helps them to integrate themselves into the society. However, sign language itself is not enough. With this boon comes a problem, the signs are often mixed up and are confusing to someone who has never learnt it or has learnt it in a different language.

The project aims at building a machine learning model that will be able to classify the various hand gestures used for fingerspelling in sign language. This requires operability in different environments with a large range of possible users, ideally under arbitrary conditions. In this user independent model, classification machine learning algorithms are trained using a set of image data and testing is done on a completely different set of data.

In this project we are presenting Sign Language Recognition System using American Sign Language. Using this system, the user will be able to use the webcam or camera to predict the sign language gestures in real-time.

We are using various images of the American Sign Language gestures and processing them with various Computer Vision techniques and creating a Machine Learning model using it.

INDEX

FIGURE INDEX

1. INTRODUCTION

Language is a vital part of human connection. It is how people communicate. By learning a language, it means you have mastered a complex system of words, structure, and grammar to effectively communicate with others. To most people, language comes naturally. We learn how to communicate even before we can talk and as we grow older, we find ways to manipulate language to truly convey what we want to say with words and complex sentences. It is so deeply embedded in our everyday routine that we often take it for granted and don't realize its importance. Sadly, in the fast-changing society we live in, people with hearing or speech impairment are usually forgotten and left out.

Why Sign Language?

Sign Language is a form of communication used primarily by people hard of hearing or deaf. This type of gesture-based language allows people to convey ideas and thoughts easily overcoming the barriers caused by difficulties from hearing issues. Sign language, although being a medium of communication to deaf people, still have no meaning when conveyed to a non-sign language user. Hence, broadening the communication gap.

Why do we need a Sign Language Recognition System?

A major issue with this convenient form of communication is the lack of knowledge of the language for the vast majority of the global population. Just as any other language, learning Sign Language takes much time and effort, discouraging to from being learned by the larger population.

However, an evident solution to this issue is present in the world of Machine Learning and Image Detection. Implementing predictive model technology to automatically classify Sign Language symbols can be used to create a form of real-time captioning for virtual conferences like Zoom meetings and other such things. This would greatly increase access of such services to those with hearing impairments as it would go hand-in-hand with voice-based captioning, creating a two-way communication system online for people with hearing issues. It will be a great tool for the people with hearing or speech impairment to communicate their thoughts and also help the non-sign language user to understand what the latter is saying.

How does Sign Language Recognition Work?

In day to day lives: Sign Language is a visual language that incorporates gestures, facial expressions, head movements, body language and even the space around the speaker. Hand signs are the foundation of the language. Many signs are iconic, meaning the sign uses a visual image that resembles the concept it represents. For instance, to express the concept of "deer" in ASL, you would hold your hands up to either side of your head, fingers spread, to represent antlers. Actions are often expressed through hand signals that mimic the action being communicated, if you wished to sign the concept "eat," you would bring your fingers and thumb of your dominant hand together as if holding food and then move your hand toward your mouth.

The alphabet is an important series of signs. Some hand signs for letters resemble the written form of the respective letter. When you use the hand signs for letters to spell out a word, you are finger spelling. Finger spelling is useful to convey names or to ask someone the sign for a particular concept. ASL uses one-handed signals for each letter of the alphabet (some other sign languages use both hands for some letters). Many people find finger spelling the most challenging hurdle when learning to sign, as accomplished speakers are very fast finger spellers.

To express an ongoing action, such as "thinking," you would make the sign for "think" twice in a row. Some signs in ASL can serve as either a noun or a verb, depending on how you sign them. In general, you'd sign a verb using larger hand gestures and a noun by using smaller gestures that are doubled. At times, this can cause confusion. The sign for "food" is the same as doubling the sign for "eat," yet the sign for "eating" is also a repeated "eat" sign. In these cases, the receiver usually knows what you mean by the context of your sentences or the size of your gestures.

Inflections: Almost any sign can be modified. Furrowing your eyebrows, tilting your head, puffing your cheeks or shifting your body are just a few ways that you can use to change the meaning of what you are saying. Any inflection that doesn't require your hands is called a nonmanual marker. An accomplished ASL speaker can convey a lot of information with only a few gestures coupled with nonmanual markers.

Another way to modify signs, particularly action signs, is to alter the speed with which you make the sign or by directionalizing the sign. If you make the sign for "eat" very slowly, for example, you can communicate that you took your time while eating. To directionalize concepts, you orient signs in a particular direction to communicate a specific meaning. If you wanted to communicate the English phrase "I gave you a gift" to your receiver, you would make the sign "give" towards your receiver, followed by the sign for "gift." There is no need to make the signs for "I" or "you," because they were understood when you directionalize the sign.

Problems Statement

Developing software applications to assist sign language communication with deaf people is an important area of work. Applications of this type could be used to aid deaf people so that they can interact with computer systems using sign language. Translation systems built on automated sign language recognition (ASLR) could assist - 3 - communication between deaf people and people who may not know sign language. Additionally, ASLR systems could be used to aid in the teaching of sign languages.

Some work has been done on ASLR, but it is not a fully mature knowledge area. Ong and Ranganathan provided an excellent survey of the ASLR state of the art, including a review of major contributions as well as an analysis of neglected areas and suggestions for future work [Ong05]. Fortunately, ASLR has many similarities to automatic speech recognition (ASR), so it may serve as a useful reference point for designing ASLR systems as well as analyzing their performance. Herein, literature relevant to ASLR and ASR is reviewed, the gaps relevant to the proposed work are identified, and further research in the area is proposed.

2. LITERATURE SURVEY

An early vision-based system for the recognition of sign language is presented in[1]. Starner and Pentland use a single video camera and uniformly colored gloves to aid the segmentation and the feature extraction processes. Later they also show, that a user-calibrated skin color model delivers similar results in a known environment.

Hienz et al[2] use color coded gloves which allow to obtain detailed information about each finger of the dominant hand. The environment is restricted to an empty white background. In arbitrary environments however, neither skin color nor any other color can be guaranteed to appear only within the object of interest, which is the hand. Thus, relying on color information only is not sufficient, not even with the aid of colored gloves.

Siming He[3] proposed a system having a dataset of 40 common words and 10,000 sign language images. To locate the hand regions in the video frame, Faster R-CNN with an embedded RPN module is used. It improves performance in terms of accuracy. Detection and template classification can be done at a higher speed as compared to single stage target detection algorithm such as YOLO. The detection accuracy of Faster R-CNN in the paper increases from 89.0% to 91.7% as compared to Fast-RCNN. A 3D CNN is used for feature extraction and a sign-language recognition framework consisting of long- and short-time memory (LSTM) coding and decoding network are built for the language image sequences. On the problem of RGB sign language image or video recognition in practical problems, the paper merges the hand locating network, 3D CNN feature extraction network and LSTM encoding and decoding to construct the algorithm for extraction. This paper has achieved a recognition of 99% in common vocabulary dataset.

In[4], a low-cost approach has been used for image processing. The capture of images was done with a green background so that during processing, the green color can be easily subtracted from the RGB colourspace and the image gets converted to black and white. The sign gestures were in Sinhala language. The method that thy have proposed in the study is to map the signs using centroid method. It can map the input gesture with a database irrespective of the hands size and position. The prototype has correctly recognized 92% of the sign gestures.

The paper by M. Geetha and U. C. Manjusha[5], make use of 50 specimens of every alphabet and digit in a vision-based recognition of Indian Sign Language characters and numerals using B-Spine approximations. The region of interest of the sign gesture is analyzed and the boundary is removed. The boundary obtained is further transformed to a B-spline curve by using the Maximum Curvature Points (MCPs) as the Control points. The B-spline curve undergoes a series of smoothening process so that the features can be extracted. Support vector machine is used to classify the images and the accuracy is 90.00%.

In[6], Pigou used CLAP14 as his dataset[7]. It consists of 20 Italian sign gestures. After pre-processing the images, he used a Convolutional Neural network model having 6 layers for training. It is to be noted that his model is not a 3D CNN and all the kernels are in 2D. He has used Rectified linear Units (ReLU) as activation functions. Feature extraction is performed by the CNN while classification uses ANN or fully connected layer. His work has achieved an accuracy of 91.70% with an error rate of 8.30%.

A similar work was done by J. Huang[8]. He created his own dataset using Kinect and got a total a total of 25 vocabularies which are used in everyday lives. He then applied a 3D CNN in which all kernels are also in 3D. The input of his model consisted of 5 important channels which are colour-r, colour-b, colour-g, depth and body skeleton. He got an average accuracy of 94.2%.

R. Rumana et al.[11] used a Sign Language Recognition Prototype which was a real-time vision- based system whose purpose was to recognize the American Sign Language given in the alphabet. The purpose of the prototype was to test the validity of a vision-based system for sign language recognition and at the same time, test and select hand features that could be used with machine learning algorithms allowing their application in any real-time sign language recognition systems.

[12] A video database of Indian sign language is created with a mobile front camera in selfie mode. The video was processed on a personal computer by constraining the computing power to that of a smart phone with 2GB RAM. Pre-filtering, segmentation and feature extraction on video frames creates a sign language feature space. Minimum distance classification of the sign feature space converts signs to text or speech. Sobel edge operator's power is enhanced with morphology and adaptive thresholding giving a near perfect segmentation of hand and head portions. Word matching score (WMS) estimates performance of the proposed method with an average WMS of around 90.58%.

The paper[13] presents a recognition system for understanding the words of home-service-related sign language. The hidden Markov model (HMM) had been successfully applied to speech signals and was chosen as a classifier. An entropy-based K-means algorithm was proposed to evaluate the number of states in the HMM model with an entropy diagram. Their database contains 11 home-service-related Taiwan sign language words and each word performed ten times, five males and five females are invited to perform such words. The recognition system was established by 11 HMM models, and the cross-validation demonstrates an average recognition rate of 91.3%.

L. C. Wang, et al.[14] proposed a model for measuring similarity between videos which content is Chinese Sign Language (CSL), vision and sign language semantic are considered for the model. Vision component of the model was distance based on Volume Local Binary Patterns (VLBP), which is robust for motion and illumination. Semantic component of the model computes semantic distance based on definition of sign language semantic. While quantizing the sign language semantic, contour is used to measure shape and orientation; trajectory is used for measuring location and movement.

K. Li, et al.[15] used a new approach to modeling transition information between signs in continuous Sign Language Recognition (SLR) and address some scalability issues in designing SLR systems. Leveraging upon hidden Markov modeling techniques from ASR, they proposed a modeling framework for continuous SLR having the following major advantages, (i) the system is easy to scale up to large-vocabulary SLR; (ii) modeling of signs as well as the transitions between signs is robust even for noisy data collected in real-world SLR; and (iii) extensions to training, decoding, and adaptation are directly applicable even with new deep learning algorithms. Evaluated on 1,024 testing sentences from five signers, a word accuracy rate of 87.4% is achieved using a vocabulary of 510 words. The SLR speed is in real time, requiring an average of 0.69s per sentence.

The study[16] developed a sign language recognition prototype using the Leap Motion Controller (LMC). This study aimed for full American Sign Language (ASL) recognition, which consists of 26 letters and 10 digits. Most of the ASL letters are static (no movement), but certain ASL letters are dynamic. Thus, this study also aimed to extract features from finger and hand motions to differentiate between the static and dynamic gestures. The experimental results revealed that the sign language recognition rates for the 26 letters using a support vector machine (SVM) and a deep neural network (DNN) are 80.30% and 93.81%, respectively. Meanwhile, the recognition rates for a combination of 26 letters and 10 digits are slightly lower, approximately 72.79% for the SVM and 88.79% for the DNN.

This paper[17] presents an extension to the Kinect SDK based on a contour analysis for the estimation of the hand position. The SDKs provided by the vendors of these devices usually lack on the needed details concerning hand movements that would be needed for an accurate hand gesture recognition implementation. These algorithms are later used to provide the creation of a gesture library that might be used afterwards.

The paper[18] reviews the sign language recognition approaches and finds the best method that has been used by researchers. Hence other researchers can get more information about the methods used and could develop better Sign Language Application Systems in the future.

3. BACKGROUND RESEARCH

This chapter presents information on topics related to Sign Language Recognition. More specifically it provides information on: what Sign Language is, what is the Sign Language recognition process and similar work.

Sign language is used for communication by community of hearing-impaired people. Since the process of learning to speak involves feedback from hearing what you say, people who are born with a hearing impairment are not able to properly speak. Therefore, they rely on other body parts to communicate. Hands for example are used to describe the shape of the object. Hands are also used to describe actions like "go from this place to that place by pointing first to source place and drawing a line visually in space toward the destination place". In addition, facial expressions convey emotions. In this way body parts serves as alternative way of communication for hearing impaired people.
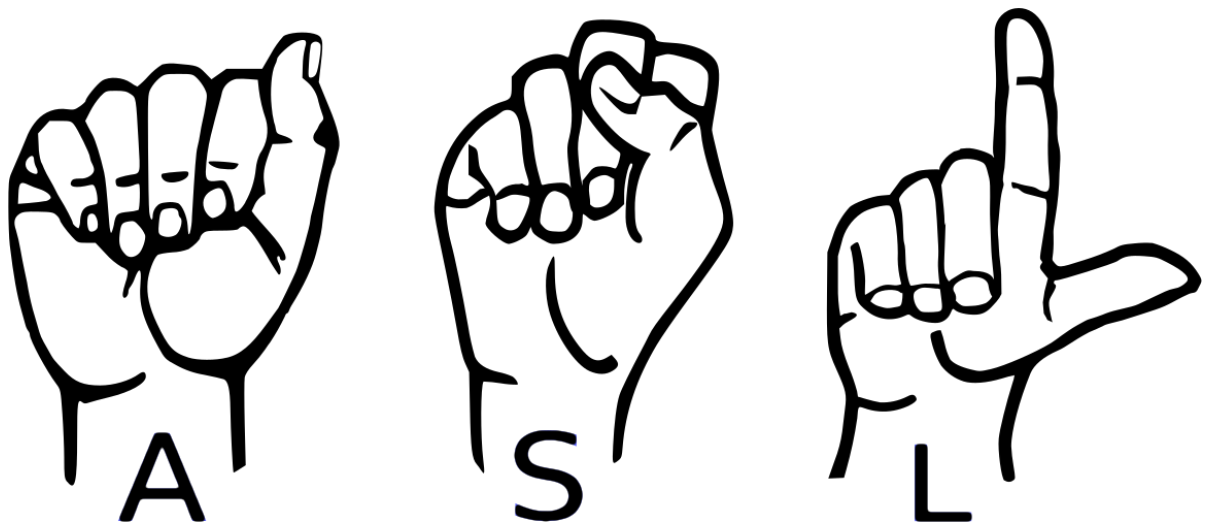
Fig. 3.1: Sign Language

Use of hands, facial expressions and others body parts to convey information forms what is known as a sign. Communication through signs is not something new for society. Babies or children use it to express their needs and feelings before they learn how to speak. Also grown people use signs for communication in specific situations, where speech is either impossible or not appropriate. Similarly, the community of hearing-impaired people use signs for communication. However, signs used by this community are different from signs used by children and normal hearing people. One of the first studies in American Sign Language (ASL) by Stokoe revealed that signs used by the HI community are structured and use rules for composition and interpretation. These rules form the language of signs or as it is better known, Sign Language.

According to the World Federation of Deaf (WFD) there are 70 million deaf people in world, whereas the world population in 2012 exceeded 7 billion. Since the community of HI people is smaller than the community of Deaf people many misconceptions exist about sign language. A common misconception is that it is universal or international language, whereas in reality hundreds of sign languages exist around the world. It is acquired by hearing impaired children as mother tongue and therefore it is unique to the culture of the hearing-impaired people. Sign Language is used to express abstract and complex meaning in addition to simple meaning (like objects in the world). Furthermore, linguistics studies have found that Sign Languages exhibit the same fundamental properties that are present in spoken languages also. For example, each sign language has its own grammar and vocabulary that can be used to express any meaning that is possible with spoken languages also. They evolve in same way that spoken languages do and they do not mimic them nor they are incomplete variants of them expressed through signs. Sign language is a natural language (as a spoken language) and in many countries it is legally recognized.

Composition:

Sign making involves the upper part of human's body. These body parts are categorized as manual and non-manual features. Manual features consist of hand's configuration and according to four hand components are used in sign making: hand shape, hand orientation, hand movement and hand location. On the other side non-manual features consist of head movements, facial expressions and body postures. Thus, sign making involves both manual and non-manual features.

Most of the meaning in sign language is conveyed through manual features, but understanding the full meaning of a sentence requires observation of non-manual features also. Non-manual features play an important role, especially in showing grammatical information. The role of nonmanual features is elaborated in more detail in grammar's section.

Each sign language has its own composition rules, but studies on sign languages have found three rules that apply to all of them:

1. **Signing space** defines the region where the signs are performed. This region is around the upper part of signer's body which is illustrated in figure 3.2.
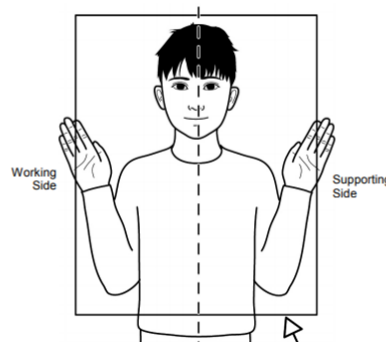

Fig. 3.2: Sign Language signing space.

2. **Symmetry** rule states that if two hands are involved in sign making and they start to move at the same time, the same hand shape must be used in both hands. This is illustrated in figure 3.3.

Fig. 3.3: Symmetry rule in Sign Language.

3. **Dominance** rules holds also for all sign languages. In analogy with writing where one of the hands is dominant (commonly used), in sign making also one of the hands is dominant. The rule states that if two hands with different shapes are involved in sign making the dominant hand is moving, while the passive hand stands still. This is illustrated in figure 3.4.
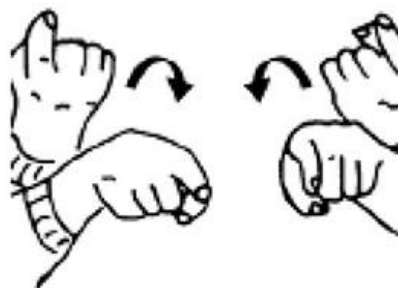
Fig. 3.4: Dominance rule in Sign Language

History of Sign Language:

The events that occurred in the history of sign language are actually pretty shocking. How deaf people experience life today is directly related to how they were treated in the past. It wasn't long ago when the deaf were harshly oppressed and denied even their fundamental rights.

The are many famous deaf people who have made a name for the deaf throughout the history of sign language and proved that deaf people can, in fact, make history.

Aristotle was the first to have a claim recorded about the deaf. His theory was that people can only learn through hearing spoken language. Deaf people were therefore seen as being unable to learn or be educated at all.

Therefore, they were denied even their fundamental rights. In some places, they weren't permitted to buy property or marry. Some were even forced to have guardians. The law had them labelled as "non-persons".

Aristotle's claim was disputed in Europe during the Renaissance. Scholars were attempting to educate deaf persons for the first time and prove the 2,000-year-old beliefs wrong. This mark in the history of sign language is what started the creation of a signed language.

Sign Language Recognition:

The term Sign Language Recognition (SLR) is used for systems aiming to automatically understand meaning of a sequence of signs from sign language without involvement of human interpreter.

In order to build such systems, the data with person signing must be obtained, then features involved in sign making must be extracted and finally combination of features must be analysed to describe the performed sign. The process is illustrated in figure 3.5.
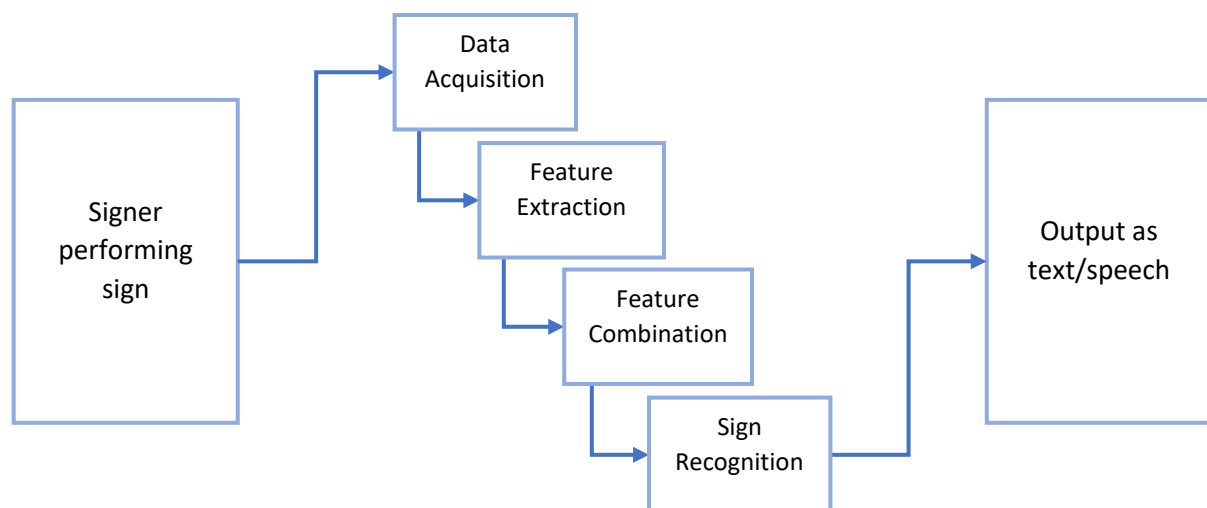
Fig. 3.5: Sign Language Recognition Process

4. PROJECT PROPOSAL

American Sign Language Recognition:

American sign language, and sign languages generally, consist of manual and nonmanual signing gestures. Manual signing consists of gestures isolated to the hands and arms. Non-manual signing encompasses broader movements of the head and torso as well as facial expressions. Manual signs can typically convey most of the lexical meaning in a sentence. Additional details are then provided by the non-manual aspects of the sign. These details may include intonation, verb tense, or intensity of action. While an ideal ASLR system would incorporate both manual and non-manual gestures, doing so is nontrivial. Especially in a mobile environment, accurate capture and analysis of non-manual signing may prove extremely difficult. Despite the constrained vocabulary afforded by this limitation, many useful applications may still be constructed using only manual signs.

The basic components of a manual sign consist of the shape, orientation, location, and movement of the hand, including both the palm and fingers. For accurate classification, an ASLR system must be able to model both the static and moving components of signs. Additionally, when signs are composed to form sentences, there is movement between the individual signs that are not part of a sign and do not actually convey any meaning. This is similar to speech in the sense that the sound of a word may be affected by the words preceding and following it in a sentence. Finally, some signs may involve positions where some fingers obscure others, or if both hands are involved, one hand may partially obscure the other. More generally, signs are not flat but exist in three-dimensional space, so this must be accounted for when capturing data for an ASLR system.

Sign Gesture Capture and Classification:

Computer vision and direct capture using gloves [27] are the primary methods of capturing hand gesture data for ASLR. Generally, computer vision is a more desirable approach because it does not require specialized equipment and the user does not have to wear a special device to use the system. As such, a large body of research exists using computer-vision based approaches. With vision approaches, the two primary concerns related to ASLR are tracking of the hands and feature extraction. In regard to hand tracking, usually the full upper body of the signer needs to be in the camera's field of view. Using 2D video from a standard camera usually requires some restriction on the background and the clothing worn by the signer. Three-dimensional video using stereo cameras can overcome many of these limitations at the expense of greater computational cost.

Critical features for detecting full signs include hand position (relative to the body), shape, and orientation. Additionally, motion trajectories of the hands may be useful for some classifications. An in-depth discussion of tracking and feature extraction can be found in the survey by Ong and Ranganathan [27].

Upon feature extraction, signs are generally classified by either taking a sign as a whole and using a single classification step, or by breaking the sign into multiple components, classifying each component, and making a final classification based on the results of the components. The primary classification methods in the literature are neural networks and hidden Markov models (HMM) [Ong05]. Both methods have been shown to yield good results, but frequently each excels at different types of signs. Neural network-based approaches are frequently well-suited to classification of non-moving signs. HMMs, which are good at classifying time-series data, are best suited to dynamic, moving signs and series of signs formed into sentences.

Questions and Objectives:

The primary objective of this thesis is to create a proof-of-concept automated sign language gesture recognition system using cloud resources, demonstrating and evaluating various architectural strategies for implementing such a system.

The prototype system was created using pixel first approach. In earlier version the system was predicting sign language by analysing every pixel of an image. Which was so much resource costly, time consuming & less accurate. Because it captures a lot of noise like other body part of signer, background, unbalanced light with the hand sign. And these make system less efficient.

That's why in the current version of the system though it captures the everything even background noise, but it takes only some key points of hand and analyse them to predict the gesture. Which makes the system lightweight as well as more efficient.

Additionally, the beautiful GUI can help anyone to manage everything easily.

Flow Chart:
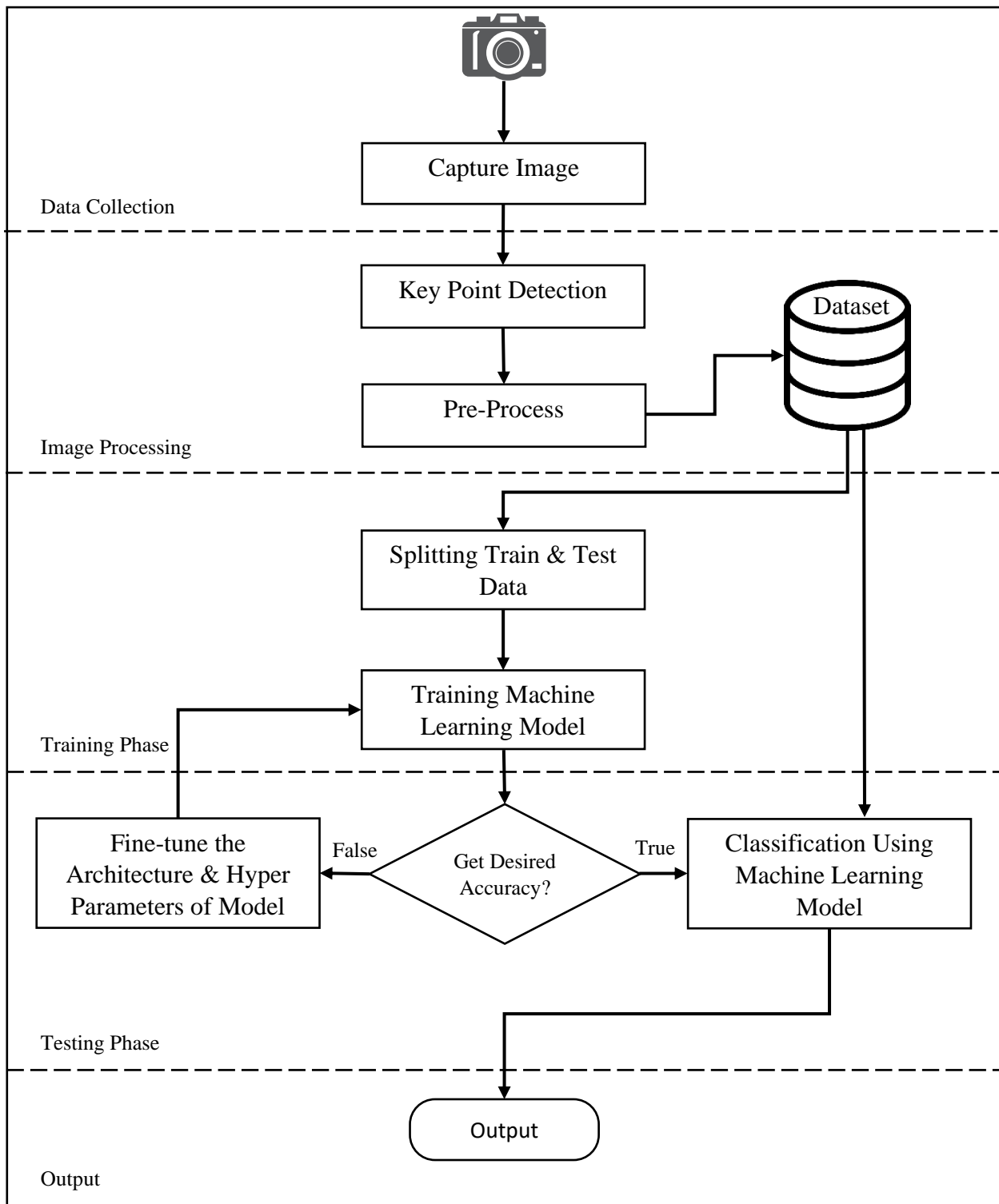
Fig. 4.1: Flow Chart of Project Overview

13

5. SYSTEM REQUERMENTS

From data collection to recognize sign & giving output the project needs some system requirements. Here's the list of both Hardware & Software Requirements. These requirements are very minimum. We could upgrade Requirements anytime.

Hardware Requirements(min):

- **Processor:** Intel core i3 / AMD Ryzen 3
- **RAM:** 8GB
- **Storage:** 5GB
- **Graphic Card:** Integrated Graphic Card
- **Input Device:** Keyboard, Mouse
- **Output Device:** Monitor, Speaker (for text to speech)
- **Capture Device:** External Webcam (if don't have in-built camera)

Software Requirements(min):

- **Python 3.10**
- **Operating System:** Windows 7 – 11 / Linux / macOS
- **IDE:** VSCode / PyCharm / any other IDE

External Requirements:

Python Modules-

- TensorFlow (2.12.0)
- Keras (2.12.0)
- MediaPipe (0.10.0)
- Scikit-learn (1.2.2)
- Pandas (2.0.1)
- NumPy (1.24.3)
- OpenCV (4.6.0.66)
- Matplotlib (3.7.1)

6. SYSTEM ARCHITECTURE

This section presents the system design from different viewpoints since all aspects of the system cannot be shown from single viewpoint. Let's have a look at the high-level architecture of the system.

Overall architecture:

Viewed from physical viewpoint the system consists of two components: the capture devices and a computer. The role of capture device is to observe the scene and give information about the hearing-impaired person that performs the signs. This information is transmitted then to computer where they are analysed and the sign meaning is revealed. This viewpoint is presented in figure 6.1.
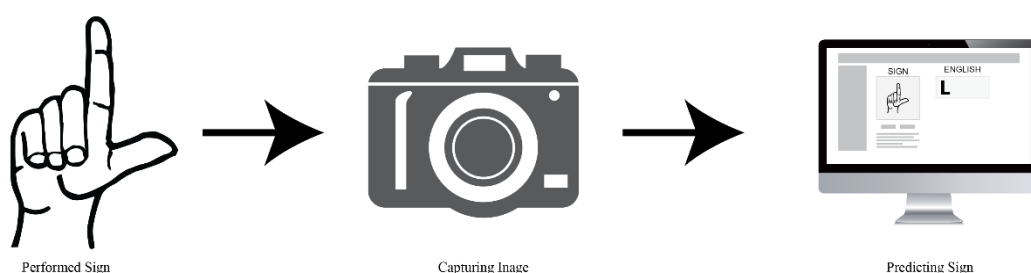
Fig. 6.1: Physical view of system

Component design:

Three main components of the system are: the presentation layer (UI), the hand component, and the recognition component. The recognition component receives the data from hand component, detects if one of the sign language components is performed.

The project investigates only manual features of Sign language. These manual features are expressed through hands. The hand component of system stores information about the shape of the hand.

This information then is accessed by recognition component that is responsible for detecting the performed sign. The recognition component checks the hand shape in order to detect the performed sign.

Since the dictionary of sign language is large, for now we are using 26 alphabets only. The main sub-component of hand component is **Hand shape recognition**.

- **Hand shape recognition** system achieved by MediaPipe. However, the shapes that it will recognize must be selected and they have to go through training phase before they are able to be recognized. Since recognition of all hand shapes is long process a set of 26 hand shapes were selected. Among selected shapes are the ones that enable recognition of basic SL components and the shape that may be used in different signs.

The last component, the presentation component is responsible for presenting the recognized sign (component) in two formats: as text and as a speech. All recognized components of sign and those of sign language are displayed on the terminal. However, since the intention is to test the recognition of each alphabet and their translation to text/speech at a time.

7. SYSTEM IMPLEMENTATION

This section presents the System Implementation of Sign Language Recognition System. This application is written in Python. In order to capture raw footage, calculating key points of hand gesture, pre-process & and store the raw data and making decision using Deep Learning the system uses a lot to tools & techniques. Here is a very low-level discussion of the system and all of its functionalities.

Hand Tracking:

In order to recognize the hand gesture at first, we need a camera feed, mainly the image of hand gesture. Here's the complete explanation of capturing image, detecting hands key points and many more. Before dive into explanation have a look at all 24 Sign Language Alphabets (Fig. 7.1).

Fig. 7.1: Alphabets of American Sign Language

Detecting Key-points:

First of all, hand gesture is captured by camera using python library OpenCV as BGR image. Then it does some minor processing to adjust image orientation, convert color mode (BGR =>RGB) and so on.

After doing all these things it goes ahead to finding some key points of every hand. With the help of MediaPipe it detects 21 specific landmarks or key points of each hand (Fig. 7.2). Each landmark is a two-dimensional coordinate (x, y). Combination of all 21 coordinates represents a shape or gesture of hand.
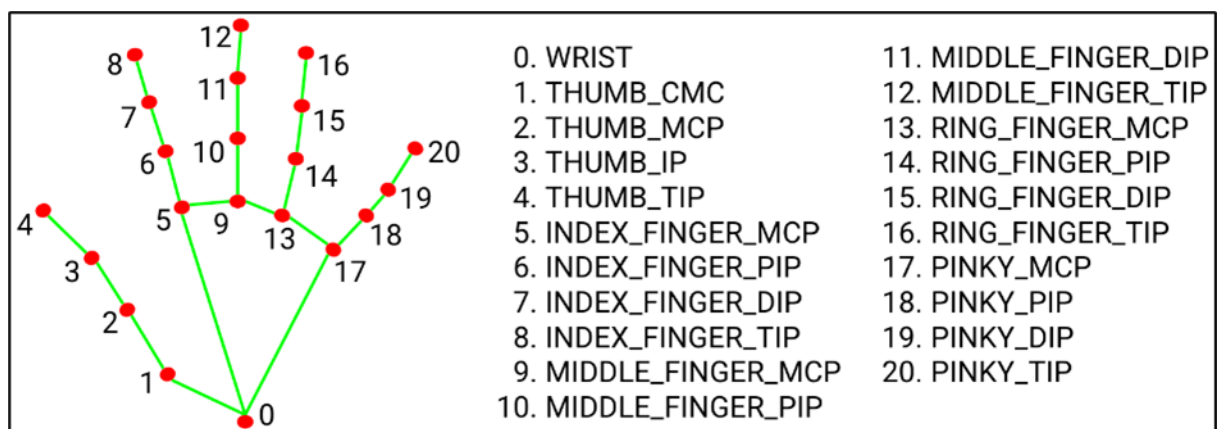
Fig. 7.2: Hand landmarks

Drawing Debug:

This is not a part of training or data collection or testing. But it's very important step to letting user know that his/her hand is detecting or not (Fig. 7.3).

Fig. 7.3: Draw Scalation & Hand Label

Calculate Landmarks:

As we know that these 21 landmarks are only things that will help us to train as well as predicting unknown gesture in future in this project, it's necessary to store them in right way. And this step is important because MediaPipe gives three-dimensional landmarks as python dictionary or JSON object. But for training the model or predicting new gesture the system needs an Array of Landmarks.

In order to do this, system iterates over all the landmarks and calculate the exact coordinates on the main image frame (Fig. 7.4). Then it converts coordinates into python list (x, y) => [x, y] and append them into another python list, which contains list of all 21 coordinates. But this is not even the right format of data to train the model. That's why it needs to process the data for one more time.

```python
def calc_landmark_list(image, landmarks) -> List:
    #: Getting image width & height
    image_width = image.shape[1]
    image_height = image.shape[0]

    landmark_point = []

    #: Keypoint
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
        #: landmark_z = landmark.z

        landmark_point.append([landmark_x, landmark_y])

    return landmark_point
```

Fig. 7.4: Calculating Landmarks

Pre-Process Landmarks:

In previous step we convert landmark dictionary into 2D list. But this still not the right format to use as a training dataset. In this step system will process the landmark list to convert 2D array into 1 dentitional array and normalized them (Fig. 7.5).

```python
def pre_process_landmark(landmark_list) -> List:
    temp_landmark_list = copy.deepcopy(landmark_list)

    #: Convert to relative coordinates
    base_x, base_y = 0, 0


for index, landmark_point in enumerate(temp_landmark_list):
    if index == 0:
        base_x, base_y = landmark_point[0], landmark_point[1]

        #: Overriding coordinates
    temp_list[index][0] = temp_landmark_list[index][0] - base_x
    temp_list[index][1] = temp_landmark_list[index][1] - base_y

    #: Convert into a one-dimensional list
    temp_landmark_list = list(itertools.chain.from_iterable(temp_list))

    #: Normalization (0 - 1)
    max_value = max(list(map(abs, temp_list)))

    def normalize_(n):
        return n / max_value

    temp_list = list(map(normalize_, temp_list))

    return temp_landmark_list
```

Fig. 7.5: Pre-processing landmark

In this pre-processing step system takes a deep-copy of original data of 2D array to work with. Then it converts all coordinates into relative coordinates so that it's become frame independent. It's doesn't mater where is the located in the frame. It will take the relative coordinates respect of wrist's coordinates.

After getting all relative coordinates it flattens all the data using NumPy. Means it converts 2-dimensinal array into one-dimensional array. And then makes all the datapoints between 0-1 value. Now this is the right format to train the model.

Data Collection:

As the system returning one-dimensional normalized array now, we can use this technique to collect the data as well as predicting new hand gesture. The system has a feature to switch between two mode i. Prediction & ii. Logging.

Fig. 7.6: Sample Dataset

Index	0.x	0.y	1.x	1.y	1.x	2.y	2.x	3.y	4.x	4.y	5.x	5.y	6.x	6.y	7.x	7.y	8.x	8.y	9.x
0	0	0	-0.2605	-0.12605	-0.48739	-0.42017	-0.59664	-0.67227	-0.48739	-0.83193	-0.36134	-0.77311	-0.36134	-1	-0.34454	-0.7395	-0.33613	-0.60504	-0.15126
0	0	0	-0.24576	-0.15254	-0.4661	-0.42373	-0.60169	-0.67797	-0.5	-0.85593	-0.33898	-0.76271	-0.35593	-1	-0.33051	-0.74576	-0.31356	-0.59322	-0.12712
0	0	0	-0.24167	-0.11667	-0.475	-0.41667	-0.6	-0.68333	-0.49167	-0.86667	-0.36667	-0.75833	-0.38333	-1	-0.35833	-0.725	-0.34167	-0.59167	-0.15833
0	0	0	-0.23729	-0.13559	-0.4661	-0.41525	-0.60169	-0.66949	-0.5	-0.84746	-0.35593	-0.75424	-0.36441	-1	-0.33051	-0.73729	-0.32203	-0.58475	-0.13559
0	0	0	-0.26271	-0.12712	-0.49153	-0.39831	-0.61017	-0.66102	-0.52542	-0.85593	-0.38983	-0.74576	-0.38136	-1	-0.34746	-0.74576	-0.33898	-0.58475	-0.16949
1	0	0	-0.15	-0.10909	-0.22273	-0.28636	-0.18182	-0.45455	-0.08182	-0.52273	-0.20455	-0.51364	-0.23182	-0.70909	-0.23636	-0.82727	-0.23182	-0.92727	-0.1
1	0	0	-0.15421	-0.09813	-0.23364	-0.27103	-0.18692	-0.42523	-0.09346	-0.50467	-0.21028	-0.50935	-0.24299	-0.70561	-0.25234	-0.8271	-0.24299	-0.92991	-0.1028
1	0	0	-0.15814	-0.09767	-0.23256	-0.27907	-0.19535	-0.44186	-0.10233	-0.50698	-0.2093	-0.52093	-0.24186	-0.71163	-0.25116	-0.82791	-0.24651	-0.93023	-0.10698
1	0	0	-0.15138	-0.09633	-0.22477	-0.27064	-0.19266	-0.42661	-0.12385	-0.50917	-0.20642	-0.51376	-0.23853	-0.70642	-0.24312	-0.8211	-0.23853	-0.92202	-0.09633
1	0	0	-0.15668	-0.09677	-0.23041	-0.27189	-0.20737	-0.43318	-0.14286	-0.51613	-0.20276	-0.51613	-0.23963	-0.70968	-0.24885	-0.82488	-0.24885	-0.93088	-0.10138
2	0	0	-0.28467	-0.07299	-0.51825	-0.19708	-0.71533	-0.24818	-0.86861	-0.35766	-0.40876	-0.64234	-0.52555	-0.90511	-0.67153	-0.91241	-0.78832	-0.84672	-0.32117
2	0	0	-0.28058	-0.07194	-0.5036	-0.20863	-0.69784	-0.26619	-0.83453	-0.38129	-0.40288	-0.64748	-0.52518	-0.89928	-0.67626	-0.91367	-0.78417	-0.84892	-0.31655
2	0	0	-0.26619	-0.06475	-0.48201	-0.21583	-0.67626	-0.27338	-0.79137	-0.38849	-0.38129	-0.65468	-0.4964	-0.91367	-0.64029	-0.92086	-0.7482	-0.84892	-0.29496
2	0	0	-0.27536	-0.06522	-0.5	-0.2029	-0.69565	-0.26812	-0.82609	-0.3913	-0.39855	-0.64493	-0.51449	-0.9058	-0.66667	-0.92754	-0.77536	-0.86232	-0.31159
2	0	0	-0.27536	-0.06522	-0.50725	-0.21014	-0.7029	-0.26812	-0.81159	-0.39855	-0.3913	-0.65217	-0.51449	-0.9058	-0.66667	-0.92754	-0.78261	-0.86232	-0.30435
3	0	0	-0.15979	-0.10825	-0.26804	-0.29381	-0.30928	-0.45876	-0.24742	-0.58247	-0.2268	-0.52577	-0.29381	-0.74227	-0.34536	-0.87629	-0.37113	-1	-0.1134
3	0	0	-0.1641	-0.10769	-0.26667	-0.30256	-0.31282	-0.46154	-0.25128	-0.58462	-0.23077	-0.52821	-0.29744	-0.74872	-0.34359	-0.88205	-0.36923	-1	-0.11795
3	0	0	-0.1641	-0.11282	-0.26667	-0.30769	-0.31282	-0.46667	-0.25641	-0.59487	-0.2359	-0.52821	-0.30256	-0.74872	-0.34872	-0.88205	-0.37436	-1	-0.12308
3	0	0	-0.15464	-0.10825	-0.26289	-0.30412	-0.31443	-0.45876	-0.26289	-0.58763	-0.2268	-0.52577	-0.28866	-0.74742	-0.34021	-0.88144	-0.36598	-1	-0.1134
3	0	0	-0.15625	-0.10417	-0.26042	-0.30208	-0.30729	-0.45833	-0.25	-0.57292	-0.22396	-0.52604	-0.29167	-0.75	-0.34375	-0.88021	-0.36458	-1	-0.10938
4	0	0	-0.27083	-0.09722	-0.4375	-0.33333	-0.40278	-0.56944	-0.23611	-0.66667	-0.43056	-0.60417	-0.5	-0.86111	-0.52083	-0.75694	-0.47917	-0.61806	-0.27778
4	0	0	-0.25874	-0.08392	-0.41958	-0.34965	-0.37063	-0.58042	-0.21678	-0.67133	-0.41958	-0.61538	-0.4965	-0.87413	-0.5035	-0.78322	-0.46154	-0.64336	-0.27273
4	0	0	-0.25	-0.09722	-0.40972	-0.33333	-0.375	-0.56944	-0.21528	-0.67361	-0.40972	-0.61111	-0.47917	-0.875	-0.49306	-0.76389	-0.45139	-0.61806	-0.26389
4	0	0	-0.25874	-0.09091	-0.42657	-0.34965	-0.38462	-0.58741	-0.2028	-0.66434	-0.42657	-0.60839	-0.4965	-0.87413	-0.5035	-0.76923	-0.45455	-0.62937	-0.27972
4	0	0	-0.25	-0.09028	-0.42361	-0.34028	-0.38889	-0.58333	-0.20833	-0.65278	-0.41667	-0.61111	-0.48611	-0.86111	-0.49306	-0.74306	-0.45139	-0.61111	-0.27083
5	0	0	-0.24051	-0.07595	-0.44304	-0.20886	-0.56962	-0.36709	-0.57595	-0.53797	-0.37975	-0.46835	-0.50633	-0.62025	-0.5443	-0.61392	-0.5443	-0.56329	-0.26582
5	0	0	-0.24684	-0.06329	-0.4557	-0.20886	-0.59494	-0.36709	-0.60127	-0.53797	-0.39873	-0.46203	-0.52532	-0.62025	-0.56962	-0.62658	-0.57595	-0.58228	-0.27848
5	0	0	-0.23899	-0.06289	-0.44654	-0.20126	-0.58491	-0.3522	-0.59119	-0.51572	-0.38365	-0.45912	-0.50943	-0.61635	-0.55346	-0.61006	-0.56604	-0.55975	-0.27044
5	0	0	-0.24359	-0.05128	-0.46154	-0.19231	-0.59615	-0.34615	-0.60256	-0.51923	-0.39744	-0.44872	-0.51923	-0.59615	-0.57051	-0.60256	-0.57692	-0.5641	-0.28205

To collect data, we need to enable Logging mode by clicking '1'. Then when we show hand gesture and press the particular key (letters only) then it captures the landmarks, go through all pre-processing steps and store them in csv file along with index of that letter.

Training Phase:

To training our model we use our own dataset, which we have created earlier. Here in this project, we used Sequential model of Keras of TensorFlow to train the Neural Network. Here's the source code of training model (Fig. 7.7).

```python
# Model Building
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((21 * 2, )),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(20, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])


# Model compilation
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
# Feeding data to the Model
model.fit(
    X_train,
    y_train,
    epochs=1000,
    batch_size=128,
    validation_data=(X_test, y_test),
    callbacks=[cp_callback, es_callback]
)
```

Fig. 7.7: Training Model Code

Implementation:

First of all, we loaded the dataset using NumPy (np.loadtxt(…)) which we stored earlier in a csv file. Then we split dataset using Scikit-Learn (train_test_split(…)).

Activation Function: Here we used 1 Input layer of 42 (21*2), which is the normalized version of all 21 landmarks. Then we used 3 Dense Layers. The first two hidden Dense layer used Rectified Linear Unit (ReLU) and the last one used SoftMax Activation Function with the unit of 24. Because at the end we need to detect only 1 class out of 24 output class (letters).

Also, we use Dropout Layer, cause The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.

```
...    Model: "sequential"
       _____
        Layer (type)                Output Shape              Param #
       =================================================================
        dropout (Dropout)           (None, 42)                0

        dense (Dense)               (None, 20)                860

        dropout_1 (Dropout)         (None, 20)                0

        dense_1 (Dense)             (None, 10)                210

        dense_2 (Dense)             (None, 24)                264

       =================================================================
       Total params: 1,334
       Trainable params: 1,334
       Non-trainable params: 0
       _____
```

Fig. 7.8: Model Summary

Loss Function: In this project we are using **Sparse Categorical Crossentropy** (SCCE). Because,

  i. Our classes are mutually exclusive.
 ii. It conserves time and memory.
iii. Mainly, it doesn't need to do sum of all samples to calculate output like Categorical Crossentropy.

$$\text{Logp} (S \in C)$$

Adam Optimizer is used for compiling the model. After compiling and fitting data to the model we need to save the model with the extension of '*.h5'.

Testing Phase

To analyse the image, it will again use the hand detection system. Again, the system will repeat the same process of data collection. But this time instead of merging landmark's array in a Dataset, it will feed every single array of each frame to the pre trained Neural Network Model to predict the appropriate class of performed sign or hand gesture.

the array will send to the Neural Network Model or the Brain of project. The brain analyses array data by compeering with prior given data used to training the brain. If it finds any match with alphabet classes then output the detected alphabet class index. Once we get the index of class, we can very easily find the alphabet.
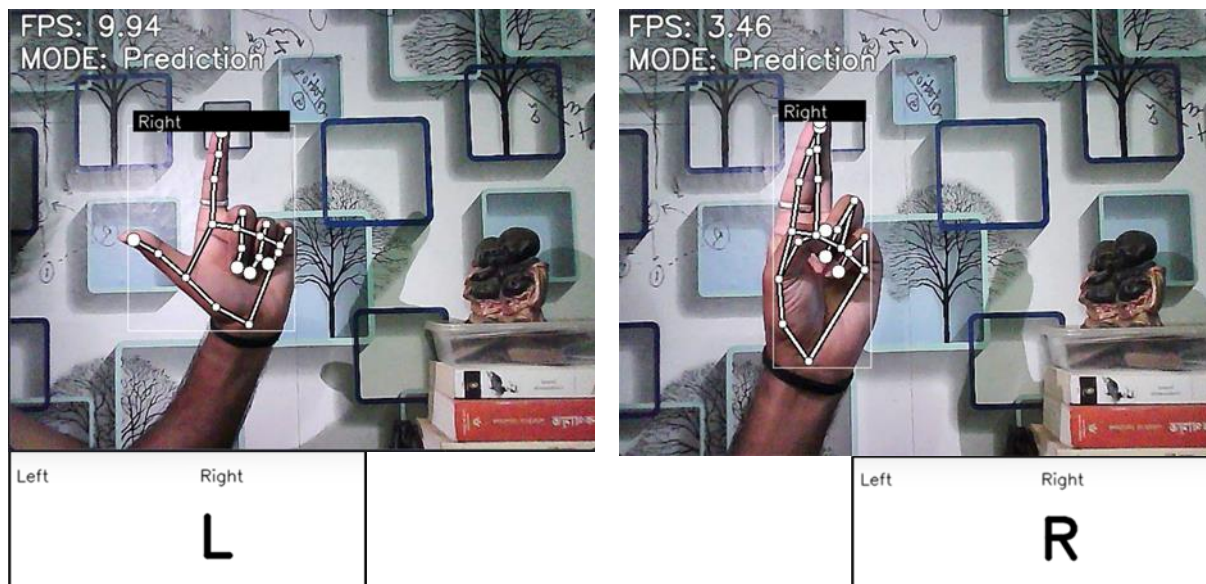

Fig. 7.9: Real-Time Testing Model

8. RESULT

This chapter presents the evaluation of the project against its objectives and selected software development process.

First objective is related to detecting hand, collecting & processing data and storing data. The system can detect hand and return location vector of hand. Then it takes the hand landmarks (key points). Captured landmarks is converted into one-dimensional array as the machine doesn't understand two-dimensional array. Also, for better accuracy it normalized all datapoint between 0 to 1 value. And finally, it stores a row of data points in a csv file(keyframe.csv).

The second objective is related with train model & recognition of class of signs or gesture from given sign in following section:

Alphabet recognition: The dataset consists of 24 alphabet letters of American Sign Language. It matches the given data in this case captured hand landmarks from live video frame with prior data or pre-trained model to predict the corresponding class. The system showed high recognition accuracy for alphabet class that performed by signer.

Discussion:

For training the model we used 2400 data. Each and every data contains 43 columns in which 1st column is the label of corresponding 2nd – to 43rd column(42 features). And in 42 featute every 2 feature is a normalized relative coordinate of hand landmarks. We trained our model by using our dataset. The training accuracy was reduced to 75% (Fig 8.1) but the real-time predictions were predominantly correct.

```
model_1.compile(
    optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = 'accuracy'
)

model_1.fit(x_train, y_train, epochs=10)

Epoch 1/10
920/920 [==============================] - 3s 2ms/step - loss: 2.5220 - accuracy: 0.2685
Epoch 2/10
920/920 [==============================] - 2s 2ms/step - loss: 1.5243 - accuracy: 0.5462
Epoch 3/10
920/920 [==============================] - 2s 2ms/step - loss: 1.1099 - accuracy: 0.6647
Epoch 4/10
920/920 [==============================] - 2s 2ms/step - loss: 0.8613 - accuracy: 0.7378
Epoch 5/10
920/920 [==============================] - 2s 2ms/step - loss: 0.6807 - accuracy: 0.8001
Epoch 6/10
920/920 [==============================] - 2s 2ms/step - loss: 0.5436 - accuracy: 0.8450
Epoch 7/10
920/920 [==============================] - 2s 2ms/step - loss: 0.4360 - accuracy: 0.8826
Epoch 8/10
920/920 [==============================] - 2s 2ms/step - loss: 0.3409 - accuracy: 0.9131
Epoch 9/10
920/920 [==============================] - 2s 2ms/step - loss: 0.2649 - accuracy: 0.9392
Epoch 10/10
920/920 [==============================] - 2s 2ms/step - loss: 0.1965 - accuracy: 0.9603
```

Fig. 8.1: Model Accuracy

Also, when we increase image quality by using good quality camera, lighting and software specifications by increasing processing power it can capture more frame per second and can process data more efficiently, then it gives more accurate prediction.

One more thing, when images were not so much processed then also accuracy was very low. It showed improvements when we process image more and play with activation functions, loss function, epoch and other parameters.

9. FUTURE SCOPE

Sign language detection systems can be further improved to achieve higher accuracy and reliability. This includes refining algorithms and models to better recognize and interpret signs, reducing errors, and increasing system robustness. Real-time sign language recognition is a significant goal for future systems. Advancements in computational power and algorithm optimization can enable faster processing, allowing for instantaneous translation of sign language into text or speech.

While sign language primarily relies on hand gestures, incorporating the recognition of facial expressions and body movements can enhance the overall communication experience. Future systems could detect and interpret these additional cues, enabling more comprehensive sign language understanding. Sign language detection systems can be integrated with other technologies, such as augmented reality (AR) or virtual reality (VR), to create immersive communication experiences. This could involve overlaying virtual sign language interpreters or facilitating remote communication through virtual environments.

Developing sign language detection systems for mobile devices can greatly enhance accessibility. Mobile apps could enable on-the-go translation and communication support, empowering individuals who use sign language to interact more effectively in various situations. Sign language detection systems can be valuable tools for teaching and learning sign language. They can provide real-time feedback, suggest corrections, and offer interactive lessons, making sign language education more engaging and accessible. Integrating sign language detection into existing technologies, such as video conferencing platforms, online content, or smart home devices, can improve accessibility for individuals who use sign language. This would allow seamless communication and interaction across different digital platforms.

Future sign language detection systems could be designed to adapt to individual users' signing styles, preferences, or regional variations. Customization options would provide a more personalized and inclusive communication experience. Sign language detection systems have the potential to empower individuals who use sign language, fostering inclusivity and breaking down communication barriers in various domains such as education, employment, healthcare, and social interactions.

10. CONCLUSIONS

In conclusion, sign language detection systems hold tremendous potential for transforming communication and fostering inclusivity for individuals who use sign language. These systems have the ability to bridge the gap between sign language and spoken or written language, enabling seamless interaction with non-signers in various settings.

By continually improving sign language detection systems, we can empower individuals who use sign language in education, employment, healthcare, and social interactions. Breaking down communication barriers and promoting inclusivity will create a more equitable society where everyone can communicate effectively and participate fully in various aspects of life.

As technology continues to evolve, it is crucial to prioritize research, development, and collaboration to realize the full potential of sign language detection systems. Through ongoing innovation and efforts, we can contribute to a more inclusive and accessible future for individuals who use sign language.

However, we were able to achieve training accuracy of 69.23% and testing accuracy of 92.36%. As future work it is intended to keep improving the system and make experiments with complete language datasets. It is also intended to test systems able to interpret dynamic sign language gestures where there is a need for reliable solutions for the problem of start and end gesture identification. As a conclusion one can say that although there is still much to do in the area, the proposed solution is a solid foundation for the development of any vision-based sign language recognition user interface system. The sign language grammar can be easily changed, and the system can be configured to train the new language gestures.

REFERENCES

1. T. Starner, J. Weaver, A. Pentland. Real-Time American Sign Language Recognition from Video Using Hidden Markov Models, M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 375, 1996.

2. H. Hienz, K. Grobel, Automatic Estimation of Body Regions from Video Images, in I. Wachsmuth and M. Fröhlich (Eds.) Gesture and Sign Language in Human Computer Interaction (Berlin: Springer-Verlag,1998), 135-145.

3. S. He. (2019). Research of a Sign Language Translation System Based on Deep Learning. 392-396. 10.1109/AIAM48774.2019.00083.

4. Herath, H.C.M. & W.A.L.V. Kumari, & Senevirathne, W.A.P.B & Dissanayake, Maheshi. (2013). IMAGE BASED SIGN LANGUAGE RECOGNITION SYSTEM FOR SINHALA SIGN LANGUAGE

5. M. Geetha and U. C. Manjusha, "A Vision Based Recognition of Indian Sign Language Alphabets and Numerals Using B-Spline Approximation", International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 3, pp. 406-415. 2012.

6. Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham.

7. Escalera, S., Baró, X., Gonzàlez, J., Bautista, M., Madadi, M., Reyes, M., . . . Guyon, I. (2014). ChaLearn Looking at People Challenge 2014: Dataset and Results. Workshop at the European Conference on Computer Vision (pp. 459-473). Springer, Cham.

8. Huang, J., Zhou, W., & Li, H. (2015). Sign Language Recognition using 3D convolutional neural networks. IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). Turin: IEEE.

9. Edon Mustafa. Sign Language Interpretation using Kinect, MSc in Software Engineering and Telecommunications, The University of Sheffield. 2014.

10. https://www.kaggle.com/ - Sign Language MNIST Dataset. (Accessed on 25th July, 2022)

11. "A Review Paper on Sign Language Recognition for The Deaf and Dumb" published by R. Rumana, Reddygari Sandhya Rani, and Mrs. R. Prema, in the year 2021.

12. G. Ananth Rao and P.V.V. Kishore, "Sign Language Recognition System Simulated for Video Captured with Smart Phone Front Camera", International Journal of Electrical and Computer Engineering (IJECE) Vol. 6, No. 5, October 2016, pp. 2176~2187.

13. T. Hseng, et al., "Recognition System for Home-Service-Related Sign Language Using Entropy-Based K-Means Algorithm and ABC-Based HMM," in IEEE transactions on systems, man, and Cybernetics: systems, vol/issue: 46(1), pp. 150-162.

14. L. C. Wang, et al., "Similarity Assessment Model for Chinese Sign Language Videos," IEEE Transactions On Multimedia, vol/issue: 16(3), pp. 751-761, 2014.

15. K. Li, et al., "Sign Transition Modeling and a Scalable Solution to Continuous Sign Language Recognition for Real-World Applications," ACM Transactions on Accessible Computing (TACCESS), vol/issue: 8(2), pp. 7-23.

16. T. W. Chong and B. G. Lee, "American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach," Sensors, vol. 18, no. 10, p. 3554, 2018.

17. T. Cardoso, J. Delgado and J. Barata, "Hand Gesture Recognition towards Enhancing Accessibility," 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Infoexclusion (DSAI 2015), pp. 419-429, 2015.

18. Suharjito, R. Anderson, F. Wiryana, M. C. Ariesta and G. P. Kusuma, "Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output," Based on Input-Process-Output. Procedia Computer Science, pp. 441-448, 2017.

19. J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

20. https://www.tensorflow.org/api_docs/ - TensorFlow v2.12.0 API Documentation. (Accessed on 26th July 2022)

21. https://keras.io/api/ - Keras API reference. (Accessed on 26th July 2022)

22. https://developers.google.com/mediapipe/api/solutions/ - MediaPipe Tasks API reference. (Accessed on 26th July 2022)

23. https://scikit-learn.org/stable/modules/classes.html - Scikit-learn 1.2.2 API Reference. (Accessed on 26th July 2022)

24. https://pandas.pydata.org/docs/ - Pandas documentation. (Accessed on 26th July 2022)

25. https://numpy.org/doc/1.24/ - NumPy documentation. (Accessed on 26th July 2022)

26. https://docs.opencv.org/4.x/ - OpenCV modules. (Accessed on 26th July 2022)

27. S. C. W. Ong and S. Ranganathan, "Automatic sign language analysis: a survey and the future beyond lexical meaning", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pp. 873–891, Jan. 2005.