

Motion Based Gesture Detection Using Frame Composition LSTM

by

Ishraqul Islam
19141008

Md. Saqif Islam
19101238

Mahin Islam Provat
19101074

Shaneen Shadman Khandakar
19101176

Fardin Junayed Karim
19101198

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2022


© 2022. Brac University
All rights reserved.

Declaration

It is with this declared that

1. While earning our degree at BRAC University, we created the thesis that was submitted.
2. The thesis does not include any already published or written by a third party content unless it is appropriately cited in the references.
3. Nothing in the thesis has been approved or submitted for consideration for any other degree or diploma at a university or other institution.
4. We have acknowledged all primary sources of help.

Student's Full Name & Signature:



Ishraqul Islam
19141008



Shaneen Shadman Khandakar
19101176



Mahin Islam Provat
19101074



MD. Saqif Islam
19101238



Fardin Junayed Karim
19101198

Approval

The thesis titled “Motion Based Gesture Detection using Frame Composition LSTM” was submitted by

1. Ishraqul Islam (19141008)
2. Md.Saqif Islam (19101238)
3. Mahin Islam Provat (19101074)
4. Shaneen Shadman Khandakar (19101176)
5. Fardin Junayed Karim (19101198)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 20, 2022.

Examining Committee:

Supervisor: (Member)



Dr. Md. Khalilur Rhaman, PhD
Professor

Department of Computer Science and Engineering
BRAC University

Program Coordinator: (Member)

Md. Golam Rabiul Alam, PhD
Professor

Department of Computer Science and Engineering
BRAC University

Head of Department: (Chair)

Sadia Hamid Kazi, PhD
Chairperson & Associate Professor

Department of Computer Science and Engineering
BRAC University

Abstract

Years of technological progress have made machines capable of identifying humans in images and videos. Moreover, machines like computers can also detect our hand gestures. Gesture recognition is the tool needed to comprehend sign languages. Sign language recognition is an important part of computer vision that uses the visual-manual modality of expression. This method solves the communication barrier between the deaf and mute and the common people. Currently, in the world, there are around 432 Million deaf mutes which is around 5% of the total global population. To solve this problem of communication gap we are focusing on creating an application for detecting sign language which will detect hand gestures and show us the output in the form of text. There are different sign languages present, but in our paper, we are mainly dealing with American Sign Language (ASL). Thus for this research, there are certain datasets present on the internet but we will be collecting our own set of words via our Real-time data collection system and make the sentences by using our model. To develop this model we are using both Long Short Term Memory.LSTM networks are a class of RNN that may learn order dependency in sequence prediction challenges. This is a necessary characteristic in complicated problem fields such as machine translation, and speech recognition, therefore we will be using it to recognize the gesture from images and video captured via the camera or webcam. Furthermore, to detect the pose and model it, we are using the MediaPipe Holistic library with the help of OpenCV. This helps us draw the landmarks on skeleton poses. Thus, giving us a generalized overview of an individual's appearance and background, allowing more focus on the perception of motion.Hence, extracting features from each frame of our videos and then composing them onto LSTM lead us into naming our model Frame Composition LSTM.

Keywords: ASL; Frame Composition LSTM; MediaPipe Holistic; Hand Gesture Recognition;

Dedication

We would like to dedicate our paper to our beloved parents and relatives and our honorable Supervisor who helped us with his contineous support Dr. Md. Khalilur Rhaman.

Acknowledgement

Firstly, all praises goes to the Almighty who blessed us with our well being for completing this thesis without any major interruption and always guiding to choose the right path. We are forever grateful to our Supervisor, Dr. Md. Khalilur Rhaman for his continuous support guide and helping us whenever we face any problem. And finally to our parents, friends and relatives with their throughout support. With their kind support and prayer we are now on the verge of our graduation. We are eternal grateful to all of them.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
Nomenclature	ix
1 Introduction	1
1.1 Motivation and Goals	1
1.2 Project Scope	2
1.3 Overview	2
2 Literature Review	3
2.1 American Sign Language(ASL)	3
2.2 Gesture Detection/Data Collection Approaches	4
2.2.1 OpenPose	4
2.2.2 Mediapipe Holistics	5
2.2.3 Radar Sensors	5
2.3 Gesture Recognition Approaches	6
2.3.1 Deep Learning YOLOv3 Model	6
2.3.2 Neural Network Using Support Vector Machine	7
2.3.3 3DCNN and LSTM with FSM Context-Aware Model	8
2.3.4 Multi-Level Feature LSTM Model	8
2.3.5 Short-Range Radar with LSTM Encoder Model	9
2.3.6 Kinect Sensor with Conditional Random Fields Model	10
2.3.7 CNN Model	10
2.3.8 Sensor Gloves Model	11
2.3.9 Research Summary	12

3	Research Methodology	14
3.1	Proposed Model	14
3.2	Dataset	16
3.2.1	Data Collection	16
3.2.2	MediaPipe Holistics For Feature Extraction	17
3.3	Data Pre-processing	18
3.3.1	Resolving Null and Missing values	18
3.3.2	One hot encoding	18
3.4	System Implementation	19
3.4.1	Recurrent Neural Network	19
3.4.2	LSTM and GRU for Gesture Recognition	20
3.4.3	Working Mechanism of LSTM	20
3.4.4	Working mechanism of GRU	22
3.5	Real-Time Gesture Recognition	23
4	Experimental Analysis & Results	28
4.1	System In Operation	28
4.2	Training and Testing Sets	30
4.3	Results	32
5	Conclusion And Future Scope	35
5.1	Conclusion	35
5.2	Future Scope	35
	References	38

List of Figures

2.1	Basic ASL Gestures	4
3.1	Workflow diagram	15
3.2	Data Collection	17
3.3	Pose Detection via MediaPipe Holistics	18
3.4	One hot encoding	19
3.5	LSTM3-chain	21
3.6	GRU cell	22
3.7	Neural Network Model	24
3.8	LSTM Architecture	25
4.1	Predicted Output	28
4.2	Results	29
4.3	Model Summary	31
4.4	GRU accuracy epoch graph	32
4.5	LSTM accuracy epoch graph	32
4.6	Hello	34
4.7	Food	34

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

α Alpha

β Beta

δ Delta

ϵ Epsilon

σ Sigma

ASL American Sign Language

BSL British Sign Language

CNN Convolutional Neural Networks

CW Continuous Wave

FMCW Frequency-Modulated Continuous Wave

HGR Human Gesture Recognition

LSF Langue des Signes Française

LSTM Long Short Term Memory

RNN Recurrent Neural Network

SFCW Single-frequency Continuous Wave

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Motivation and Goals

Communication defines the concept of imparting or exchanging information via different mediums. Humans have attempted to communicate and express their thoughts and emotions from the dawn of time. As a result, individuals begin to express themselves through their languages, and over time, they develop signs or alphabets to carry out their purpose. As the world's population grew, individuals began to separate by adopting different identities, relocating to other locations, etc. There has been a report by Arkansas State University [7] that there are three types of communication, and they are verbal, nonverbal, and visual. These types of transmission between people have been going on for a long time. Thus, there are people with physical disabilities such as deaf, mute, etc. Thus they must use specific methods of communication, such as hand signs. Since the whole population belongs to different parts of the world, they have their specific mother tongue languages to convey their messages. Throughout this set of dialects, English is the most structured and has been used throughout the world for a very long period. So, regarding people with physical disabilities, American Sign Language (ASL) is the most used method of conveying information to each other.

One of our friends' families experienced a great deal when one of the relatives was both deaf and mute. As a result, communicating with him in sign language was quite tricky. To acknowledge him, they had to sit down with a pen and paper and write down their concepts, which can sometimes be quite difficult to understand. However, it is not possible for an individual to always carry a pen and paper with them, so this difficulty in communication inspired us to understand the idea of hand motions and produce something useful for this kind of demographic. This idea of hand motions and gestures to acknowledge a language is known as sign language. Sign languages are languages that communicate by the means of visual-manual modality. This form of communication are being used by hearing and speaking impaired people.

According to [25], no single sign language exists worldwide. In various nations or locations, there are many sign languages in use. For instance, British Sign Language (BSL) differs from American Sign Language (ASL), and Americans who are familiar with ASL may not be standard with BSL. ASL characteristics are included in some nations' sign languages. As per [25] people are not sure where ASL was created from but some theories suggest that it was developed "200 years ago" and has some common features with the French Sign Language (LSF, or Langue des Signes

Française). Even though our model can be used to recognize other sign languages, we have decided to focus our work on ASL. As per [30], the main reason for using ASL is because it has a complete natural language with its own grammar, vocabulary, and composition. The most frequently used sign language in the world is American Sign Language. Signers come from the United States and Canada, as well as sections of Mexico, Africa, and Asia.

Therefore, the purpose of our research is to take a feed from live video and detect the motion through MediaPipe Holistic and Frame Composition LSTM (Long Short Term Memory), where LSTM is a unique form of RNN (Recurrent Neural Network). Our research's sole goal is to aid hearing and speech-impaired people so they can communicate efficiently.

1.2 Project Scope

There has not been much work on motion or gesture detection let alone American Sign Language recognition using a regular camera found on day to day smart phones or laptops. This is why we are making an effort to come up with a unique framework that is able to carry out all sorts of motion detection in real-time; however, we are solely focusing on acknowledging hand gestures to understand American Sign language. Hence the system we are proposing allows users to detect hand gestures in real-time and translate them as text output using nothing else other than their smartphones or web cameras found on laptops. The model will allow users to achieve data collection in real-time from their regular cameras and then our frame composition LSTM learns the corresponding words related to their respective sequence of hand gestures. ASL to American English translation can then be portrayed on the screen in real-time from their complementing gestures without the use of any complex hardware.

1.3 Overview

Our research work comprises a deeper study of the techniques used for motion gesture detection which focuses on sign languages such as (ASL) American Sign Language. This project not only focuses on a new approach but a robust framework which we are calling Frame composition LSTM that works in real-time. In Chapter 2, we are reviewing the discrete types of models and detection systems used by the different hand gesture detection papers, and why we are choosing the key components of our model. In Chapter 3, we are going over the proposed system, methodology, data collection, and the working mechanism of our algorithm. In Chapter 4, we are covering our train-testing process and show our results. Lastly, in Chapter 5, we are concluding our research and giving a brief on the future aspects of our work.

Chapter 2

Literature Review

Hand gesture detection and recognition have been a huge keen interest in the modern era. Related works covering this topic have already been published, and while a comprehensive review of this field may be quite challenging, we will focus on the recent developments that are relevant to our research topics.

2.1 American Sign Language(ASL)

According to [2], the Survey of Income and Program Participation (SIPP) which is one of the few nationwide surveys that consistently gathers information about the number of Americans who are deaf or have hearing loss, their survey shows that approximately 1 in 20 Americans are currently deaf or hard of hearing. There are roughly 10 million people who have hearing loss, and there are about 1 million people who are functionally deaf. ASL is a natural, complete language in the visual-spatial modality of sign language. Mainly the linguistic characteristics the spoken language is considered. ASL includes regional variants, just like other languages, and thus around the world, there are various sign languages. According to [3], American Sign Language (ASL) is the primary language of about 500000 Deaf individuals in Deaf groups and Deaf families throughout the U.S alone. The grammar of sign languages is distinct from that of spoken languages. For instance, ASL and English are two separate languages with their own vocabulary, grammar, and other features. Five parameters are used in American Sign Language (ASL) to define how a sign acts inside the signer's space. The variables involved are "hand shape", "palm orientation", "movement", "location", and "expression/non-manual" signals. For our research, we will focus on using gestures to represent full sentences (as done in real-life scenarios) rather than singular words or alphabets to make up a sentence.



Figure 2.1: Basic ASL Gestures

2.2 Gesture Detection/Data Collection Approaches

2.2.1 OpenPose

An individual's orientation in a certain manner is determined by a pose skeleton where a set of data points that connect together to describe a pose. Each data point of the skeleton can be considered as either a coordinate or a point. Thus according to [26], OpenPose is a library that is not only capable of detecting such skeletal poses but also able to act in real-time and detect multi-person human poses.

As per [27], out of the numerous features, some of the core features of OpenPose are:

- 70 facial landmarks
- 21 hand landmarks – each hand
- 15, 18, 27 body/foot landmarks

The OpenPose library first extracts characteristics from an image by utilizing the initial few layers. The collected characteristics are then sent into two parallel convolutional network layer divisions. The leading division predicts a series of 18 confidence maps, each of which represents a different portion of the human posture skeleton. The following branch predicts another set of 38 Part Affinity Fields (PAFs) that

represent the degree of relationship between parts. Lastly, the latter steps are used to tidy up the predictions given by the branches.

2.2.2 Mediapipe Holistics

According to [24], Mediapipe Holistic, like OpenPose, is one of the pipelines that has optimized face, hands, and posture components that enable holistic tracking, allowing the model to recognize hand and body positions as well as facial landmarks at the same time.

As per [27], out of the numerous features some of the core features of MediaPipe are:

- 468 facial landmarks
- 21 hand landmarks -each hand
- 33 body/ foot landmarks

The name given to the MediaPipe perception pipeline is a graph. Here, a stream of photos is fed as input, and thus landmarks are generated. The nodes in the MediaPipe Graph are called Calculators, and the edges are called Streams. Lastly, every stream has a series of packets with escalating time stamps.

2.2.3 Radar Sensors

This paper [20], gives a comprehensive survey and analysis of HGR through radar sensors. Different available radar technologies have their own similarities and differences, leading to data acquisition and representation of HGR. Thus radar telecommunications can be summarized into 2 categories based on their signals being transmitted naturally , according to [10]:

- Pulsed Radar
- “CW” Radar

“CW radars” are further categorized as “FMCW radars” and “SFCW radars” are also known as “Doppler radars”. Furthermore, different motion gestures are detected using reflected “WIFI signals”. Hence, efforts were made to develop HGR algorithms for existing sensors rather than establishing new hardware.

”Pulsed Radars”

“Pulsed radars” has an extensive spectrum of frequency from which signal impulse are transmitted. The Ultra-Wideband (UWB) communications system is referred to as the pulsed signals transmission and reception system. Making them portable, these radars have lower “Power Spectral Density” (PSD) and minimal radio frequency (RF) or microwave components. Hardware mainly consists of transmitting and receiving antennas, a “Phased Lock Loop”, a “Power Amplifier” (PA), ”quadrature mixers” and a “Low-Noise Amplifier” (LNA). Now, for its function, signal gets reflected from several different paths when transmitted by the “UWB Radar”. Using an “analog-to-digital converter” (ADC), the receiving echoes are modified. Reflected signals, however, contain unwanted information reflections known as clutter. Thus,

a simple loop-back filter is proposed for clutter removal operations. Further exploitation of pattern-recognition techniques is being applied to recognize desired hand gestures. Other filters include “Singular Value Decomposition (SVD) filter” and “Moving Target Indicator (MTI) filters”.

CW Radars

Here, the two main types are briefly explained, as detailed by [20]

SFCW Radar As detailed,[20], these radars mainly detect targets when there is a change in frequency between the signals being sent and received . Here, the shift in Doppler frequency ranged within several hertz is measured due to the motion of hands and fingers. Specified hardware for the Doppler radar designed for hand-gesture acquisition is proposed of a “low-frequency crystal oscillator”, together with a “Local Oscillator” (LO), “band pass filter”, a “power amplifier”, and a “transmitting antenna”. Later, “I phase” $I(t)$ and “Quadrature-Phase” $Q(t)$, is obtained as the input by the reflections from the hand which produce receiving signals, that are furthermore demodulated and passed through the “band-pass filter”

FMCW Radar Varying frequency signals are sent by these radars as explained by [20] . Firstly, signal is sent with a low frequency and, as time passes by, this gets increased continuously to a particular bandwidth span. Chirp time is measured as the time duration from low to high frequency. Typical “FMCW radar hardware for hand-gesture sensing consists of a front end with a “BGT24MTR12 signal generator” and “receptor chip” from a company named “Infineon” technologies, and a “Cyclone-3 FPGA” chip developed by “Altera” technologies. A center frequency of 24GHz is proposed for the K-Band for this radar. f_{min} , f_{max} , and bandwidth B are measured for the transmitted signals, and the corresponding receiving signals will have a delay factor to sense targeted movement.

Finally, all the scattered radar signals are saved properly into several formats for the signal-processing algorithms to work upon. These formats are expressed as Time-Amplitude, Range-Amplitude, Time-Range, Time-Doppler, Range-Doppler frequency, and Time-Frequency.

2.3 Gesture Recognition Approaches

2.3.1 Deep Learning YOLOv3 Model

YOLO is considered as a part of CNN (convolutional neural network) that is not only efficient but also capable of working well for object detection in real-time as per [23]. In addition to aiding with feature extraction, this also assists in gesture interpretation and the detection of potential objects of interest.

When the dataset is collected, YOLO annotation is labeled onto it, which is later fed onto the DarkNet-53 model for further configuration. In order to get the required output for the network, YOLO’s procedure uses a formula that takes hold of different coordinates, such as “PW”, “ph”, “tx”, “ty”, “tw”, “th”, “cx”, and “cy”, and uses

them as bounding box dimensions. This paper [23], gives a deeper understanding of how the algorithm works. When the image is passed onto the YOLO network, the input layer comes first and does its job, and then before passing into the hidden layer it is rendered. Then Hidden layers are greater not only in number but also in size and are also interconnected convolutions. This is followed by further extraction of data required to recognize gestures, which is done by passing it onto the feature map prediction block. Once the prediction is done, the data is then sent into the decoding section. In this section, the output predicted is mapped into an image and presented. Finally, the confidence threshold was determined to be greater than 0.5, and defined the activation according to the stride and the pad. For the YOLOv3 model, the mask was set to 0.5, the learning rate to 0.001, and the value of jitter to 0.3.

The resulting accuracy that YOLOv3 produces for training and testing is 97.68% followed by 96.2% respectively. However, the following model was proven to detect static gestures, and error was shown when there was a transition from one gesture to another.

2.3.2 Neural Network Using Support Vector Machine

In paper [9], the authors presented a system that converts the alphabets of the Bengali Sign Language to text using Neural Networks and Support Vector Machine algorithm. Despite being one of the most commonly spoken languages in the world, Bengali, very little research has been done on BdSL gesture detection to aid the deaf and mute members of the community. The dataset used in their paper was generated by taking images of different hand gestures using Microsoft Kinect.

They used a five-layer Neural Network model to detect the alphabets of the BSL. Apart from the input and output layers at the start and end, their model contains hidden layers in between, designed to carry out specific functions in order to produce the final result.

The whole alphabet predicting process starts by taking an RGB image of the hand gesture using Microsoft Kinect. This colored image is converted to grayscale and then further changed to a binary threshold image. A contour detection algorithm is run using the binary image to determine the position of the wrist. Pixels are read from left to right, starting from the bottom line and moving all the way up to the top line. The locations where pixels change from black to white and from white to black are stored. The least distance between these two points is deemed to be the wrist. To detect the fingers and joints' layout of the hand, a second algorithm is used. This works by detecting every pixel of the image and comparing the upper indexes with the lower ones. The process continues for all columns until every visible finger is identified. The joints of the fingers are also determined by using the human body ratio and the fingers' heights. Vectors found from the previous steps are then passed into the SVM algorithm. The image is matched with the training data present in the model. Finally, the matched image's corresponding alphabet is displayed as output.

To test the model’s accuracy, the researchers chose 4 letters and fed 22 gestures each as inputs. Different letters displayed different accuracy scores with ‘Shorey-O’ showing the highest value(90.90%) while ‘Kho’ produced the lowest(77.27%). The 4 letters averaged accuracy of 84.11% overall.

2.3.3 3DCNN and LSTM with FSM Context-Aware Model

As per [11], the author here portrays the use of dynamic hand gesture recognition in Smart TV Technology due to the sudden increase of features that are now being implemented into television. All these were very rare but now it is very easy to do such things. Here it shows how using ”3D CNN and LSTM with FSM Context Aware model” will allow to get accuracy of higher percentage in recognition of real-time hand motion detection.

To begin with, they first divided some individuals into a group where all of them are right-handed so that they can reduce the anomaly and they all were asked to perform some gestures which contain 120 frames. According to the writer, they recorded around 24 gestures six times from 20 different people. This recorded data was passed through a RealSense ”SR300 depth camera” where the information is a combination of ”RGB” and ”Depth” data. To extract the feature from this information they decided to run it through a CNN algorithm named 3DCNN which was capable of extracting temporal characteristics while preserving the spatial aspects of the images in the fields of motion detection and video classification.

Since the result mostly consist of 32 to 50 frames per gesture this “3DCNN model” may not be a useful in learning this process. As a result, another network is required to learn long-term temporal properties. It was suggested to combine the “3DCNN algorithm with the LSTM network” to aid in learning the characteristics of long-term time scales. The LSTM can learn long-term dependencies thanks to its intricate structure, which comprises input, output, and forgets gates that control the long-term learning of sequence patterns. Finally to make the work more accurate and recognizable in real time they decided to use the FSM Context-Aware Model which provided an accuracy between 99.2 % and 99.4 %.

2.3.4 Multi-Level Feature LSTM Model

In the research paper [14], the authors illustrate a dynamic hand gesture recognition model where their study builds robust hand shape features from two modalities which are depth and skeletal. They have then proposed the use of a multi-level feature LSTM with Conv1D, Conv2D pyramid, and lastly the LSTM block to deal with the diversity of hand features.

In this work, they have divided the hand posture feature extraction into three parts which are handcrafted skeleton features that represent global motion and specific changes to palm coordinates, joint point-cloud features that facilitate the “3-dimensional geometric transformation” feature from “point-cloud”, and finally, depth-shape features which play the role of obtaining feature vectors that represent

detailed info on the hand form.

For their “Depth-shape” model, they have selected ”FingerPaintDataset” as their suitable dataset to work which included 5 performers who used almost static hand poses. They have selected the “Dynamic hand gesture” dataset for their hand gesture recognition, which consists of “skeleton” and “depth” data. These are useful for their procedural methods. This dataset consisted of 20 performers and each performer used 5 gestures in two different ways thus summing up to a dataset consisting of 2800 sequences total.

Their network architecture uses a temporal series of features derived from gesture data which is processed using the first LSTM layer. To leverage temporal-spatial coherency, the “encoding feature vector” supplies the “Conv1D pyramid” block, the “Conv2D pyramid” block, and the “LSTM block”. In order for categorizing the motions, all characteristics from the building blocks are combined before adding them to the dense layers, thus allowing the researchers to achieve an accuracy of 96.07% for depth Data and 94.40% for Skeletal Data.

2.3.5 Short-Range Radar with LSTM Encoder Model

The Authors in [12], proposed a real-time detection system for hand gesture using an “FMCW radar soli” module which was developed by Google. Their model consists of a signal processing part that generates RDM (“range-doppler map”) sequences and their temporal properties are learned using an LSTM encoder.

This paper describes that their Radar system for Gesture recognition consists of three core items. Firstly, signal processing where information from the waveform of the FMCW radar consists of the time delay of an object approaching or moving away from the radar and its doppler shift. Secondly, Clutter extraction where Clutters created by reflection from other objects other than the hand is removed from raw RDMs before identifying motions. The clusters that may vary over time are efficiently retrieved by constructing an adaptive background model based on the GMM (“Gaussian mixture model”). Lastly is gesture detection where a CFAR (“constant false alarm rate”) method is utilized along with EWMA (“exponentially weighted moving average”) for calculating the moment average.

The dataset being used here includes a data collection of ten kinds of hand gestures: “sliding left to right”, “sliding right to left”, “rotating clockwise”, “rotating counter-clockwise”, “push”, “double push”, “drawing X”, “drawing reverse-X”, “hold”, and “double hand push”. Here the participants include 8 males and 2 women between the ages of 23 and 35, thus a total of 4000 hand gestures were obtained for this research.

Furthermore, their proposed LSTM encoder was compared with two different types of machine learning techniques, ”RNN encoder” and ”2D CNN” where their own LSTM model showed an accuracy of 99.10% and the other models showed 95.39% and 92.90% respectively.

2.3.6 Kinect Sensor with Conditional Random Fields Model

In the article [5] the author proposes a sign language detection method that uses 3D-depth information obtained from the hand motions which are generated by Microsoft's Kinect sensor and lastly applies a CRF (Conditional Random Field) to recognize the signs.

The hand tracking component in the Kinect Windows software development kit detects facial and hand locations reliably. The skeleton model is made up of ten feature points drawn from the upper body whereas the hand region is calculated by means of segmentation which is done by users wearing a black wristband. In order to identify the black wristband, RANSAC ("Random sample consensus") is used thus normalizing the detected hand shape. Using the identified hand and facial areas, features are retrieved in 3D and 2D space respectively.

To recognize sign language the author also uses, a hierarchical CRF architecture is utilized. A threshold model "T-CRF" is utilized in the first phase to differentiate between sign and non-sign patterns. Non-sign patterns are specified by the label "N-S" in this phase, whereas signs are determined by the labels in the vocabulary. The maximum entropy concept is based upon "CRF" parameter learning and also weights from the "CRF" are used to create the "T-CRF". Although the hierarchical "CRF" is good at identifying hand motions, it struggles to distinguish between different hand forms thus BoostMap embeddings are used to recognize the hand shapes.

The author's technique makes use of a dataset including 17 different hand shapes, 864 pictures are created for each hand sign. As a result the model received an accuracy of 90.4% .

2.3.7 CNN Model

The author of the article[16]proposes a hand gesture recognition model that uses CNN (convolutional neural networks) classification. Using mask images, the image in which the hand's region is separated from the rest of the part of the image. The methodology here claims the usage of the CNN classification method where the entire process flow consists of "capturing hand image", "hand ROI segmentation", "fingers segmentation", "Normalization" and lastly the "Fingers recognition".

The CNN architecture employed in this study for hand gestures classification is made up of convolutional filters that conduct kernel multiplication with the input image "(7 * 7 size)", pooling, and fully connected layers. Five convolutional layers and one fully connected layer with 1024 units comprise the proposed CNN architecture. Each convolutional layer functions as a feature extractor, extracting separate feature sets from the input source picture for classification. By combining neuron components received from convolutional layers the feature map is built. The "pooling layer" has the functionality to minimize the feature map's spatial resolution produced by convolutional layers, thus two pooling techniques are used here.

The fully connected layers carry out the evaluation of these feature representations and execution of high-level reasoning following the convolutional and pooling layers.

The CNN classification method generates eight distinct classes, each of which represents a unique hand motion. However, the overall methodology achieves an overall accuracy of 91.6%.

2.3.8 Sensor Gloves Model

The Authors in [32], portray the use of Sensor gloves to recognize Sign Languages. According to this study, the gloves transform American Sign Language (ASL) gestures into groups of 24 English alphabets and two punctuation marks, which are then translated into phrases in English.

In [32], the reason for them choosing ASL is because it is a grammatically complete natural language and the most commonly spoken language in the world. To achieve this task they decided to use data gloves made of cloth and it is fitted over the camera to recognize the gestures. Here Khan, N.Y. and Mehdi, A.S completed the task by using 7 sensor gloves from "5DT company". Each of these sensors had a specific purpose; five were used to detect the motion of the fingers and thumb, one to assess head tilt, and the other to monitor hand rotation. Optic fibers are attached to the glove in order to evaluate the flexibility of the fingers and thumb. An integer number between 0 and 4095 is returned by each sensor. This number indicates how twisted the sensor is. 4095 signifies totally bent, whereas 0 implies fully stretched. So, our input is a set of $7 * 4096$ combinations.

Furthermore, in [32], the output for a particular input pattern is calculated using the feed-forward method. The backpropagation technique is used to teach the network new information. The network employs three layers of nodes. The input layer, which is the first layer, receives seven sensor readings from the glove's sensors. So there are 7 sensors in this layer. This layer just sends the values forward without performing any processing. The data that is being received by the gloves are then passed through an "Artificial Neural Network (ANN) with feed-forward and backward propagation algorithms". The values from the preceding layer are given weights by the hidden layer, which comes after the input layer. This layer has 52 nodes in it. The output from this layer is sent to the third layer.

Output layer which is also the third layer, applies weights to the data it receives from the hidden layer as input. This layer has 26 nodes. One alphabet of the sign language subset is represented by each node. The final output is sent out by this layer. The output's last stage is applied using a threshold. Only values greater than this level are taken into account. No letter is outputted when one of the nodes produces an output over the threshold value. Four samples are taken per second. For the sign to be recognized, the user needs to continue doing it for 3/4 of a second. For more rapid performance, this cap may be decreased hence the software showed an accuracy of 88%.

Table 2.1: Research Summary Table

Author Name:	Models:	Sensors:	Accuracy:	Real-Time:
A. Mujahid, M. J. Awan, A. Yasin, et al. [23]	Yolov3, CNN, DarkNet-53	Camera	96.20%	NO
S. F. Hasan, T. M. Rahman, A. R. Chowdhury, and A. Biswas [9]	SVM	Microsoft Kinect	84.11%	NO
N. L. Hakim, T. K. Shih, S. P. Kasthuri Arachchi, W. Aditya, Y. C. Chen, and C. Y. Lin [11]	3DCNN, LSTM, FSM	depth camera	99.20%	YES
N. T. Do, S. H. Kim, H. J. Yang, and G. S. Lee [14]	LSTM, Conv1D, Conv2D	Depth Camera	96.07%	NO
C. JAE-WOO, R. SI-JUNG, and K. JONG-HWAN [12]	LSTM, RDM, GMM	Radar	92.90%	NO
H. D. Yang [5]	CRF, RANSAC, T-CRF	Kinect	90.40%	NO
P. S. Neethu, R. Suguna, and D. Sathish –Neethu2020efficient”	CNN	Camera	91.60%	NO
S.A. Mehdi; Y.N. Khan [21]	ANN	Sensor Gloves	88%	YES

2.3.9 Research Summary

The figure 2.1 above, shows a brief of some of the researches that were carried out over the years where each of them have used a variety of methods, models and detection hardwares for hand gesture recognition system. As per the table we see that the use of models range from Yolov3, CNN, and SVM to 3DCNN, CRF, and FSM. As for system detection hardware according to [23], a camera is not only cheap compared to the other types of recognition devices but also is easily capable of distinguishing different shapes, and colors of discrete objects in high resolutions. On the other hand, devices such as Microsoft’s Kinect used in the article [5]’s model rely on an infrared depth sensor that is capable of tracking 48 points of movement on the human body, thus the model achieves an accuracy of 90.4%. Also, the other model in [21] uses a sensor glove which is quite complex to not only set it up but also to use as it has to be constantly connected to numerous wires. Despite all the complexity it manages to achieve an accuracy of 88%. Lastly, for the radar system model of [20], there are numerous disadvantages since radar technology relies on the detection of a transmitted signal being reflected from an object. Hence, issues such as reflection cluttering and a vast amount of background noise can be faced. In this model accuracy of 92.90% to 95.39% was achieved by considering that the environment was in perfect condition where none of the issues took place. Therefore, in the previous research works according to the summary table above, only sensor gloves in [21] and depth camera in [11] were able to achieve real time hand gesture detection.

For using a regular camera, a reliable library that accurately carries out skeletal pose detection was needed such as OpenPose or MediaPipe. According to the tests run in [31], different circumstances such as “default conditions”, “profile angle view”, “motion speed”, “light intensity”, “hand-hand overlap” and “hand-face overlap” were used to find which library between MediaPipe and OpenPose provides better accuracy. Both MediaPipe and OpenPose performed admirably in the test. When displaying relatively static pictures, both methods have high accuracy in recognizing human body landmarks. However, for dynamic videos, OpenPose takes more time and more computing power compared to MediaPipe. Lastly, for disruptions such as motion blur, or an increase in speed of motion, MediaPipe did a better job in dealing with them. Therefore, MediaPipe is a better Skeletal pose detection Library than OpenPose.

On an average LSTM in [12], [14], and [11] provides better accuracy for hand gesture recognition where each of them achieved an accuracy of 92.9%, 96.07%, and 99.20% respectively, thus compared to the other models such as SVM (Support Vector Machine) in [9] achieved an accuracy of 84.11%. Whereas, the CNN(Convolutional Neural Network) model of [16] had an accuracy of 91.6%. On the other hand, the

Yolov3(You only look once, version 3) model in [23] got an accuracy of 96.2%. Lastly, for CRF (Conditional Random Field) model in [5] achieved an accuracy of 90.4%.

Chapter 3

Research Methodology

3.1 Proposed Model

Our proposed model would work with real-time hand gestures. Firstly, our model must be trained to know the difference between the gestures. Thus, to train our model we must first collect the data. In our case, the data that we are collecting are coordinates of certain points in the hand, face, and posture. For this purpose, Mediapipe Holistics will be used. Using MediaPipe we obtain 21 landmarks or coordinates from each of the hands and 468 landmarks from the face and lastly, we obtain 33 landmarks from the pose. Each of the landmarks contains 3 coordinates which correspond to the x,y, and z-axis. These are data collected for each frame, however, as we are aiming for real-time gesture detection we will take in 30 frames for every gesture, each of which would be repeated 30 times. Before passing on these data to form a neural network model, we must first preprocess the data by adjusting for null values and other discrepancies that might cause the model to perform in an unintentional way. After we have made the model using LSTM, we manually test the model by using openCV to provide inputs in real time. From the OpenCV feed, we collect the recent 30 frames and join the collected landmarks of the frames into an array which is then passed to our model which then matches with our given list of gestures. The output of the corresponding gesture, done by the user, is shown on the window popup of the OpenCV. For our model to provide an output, it has to match an accuracy of 70% to be shown as an output; this process is also quite fast as LSTM only has a time complexity of $O(w)$, where w is the number of weights [8].GRU is also used to replace the LSTM model to compare the two RNN's to find out which model fits proposed model better. Our proposed model identifies the hand gestures in real-time and is also very efficient as it only requires little time to process and identify the gestures that are made, thus letting the user communicate seamlessly. The flowchart 3.1 shows how our proposed model would function.

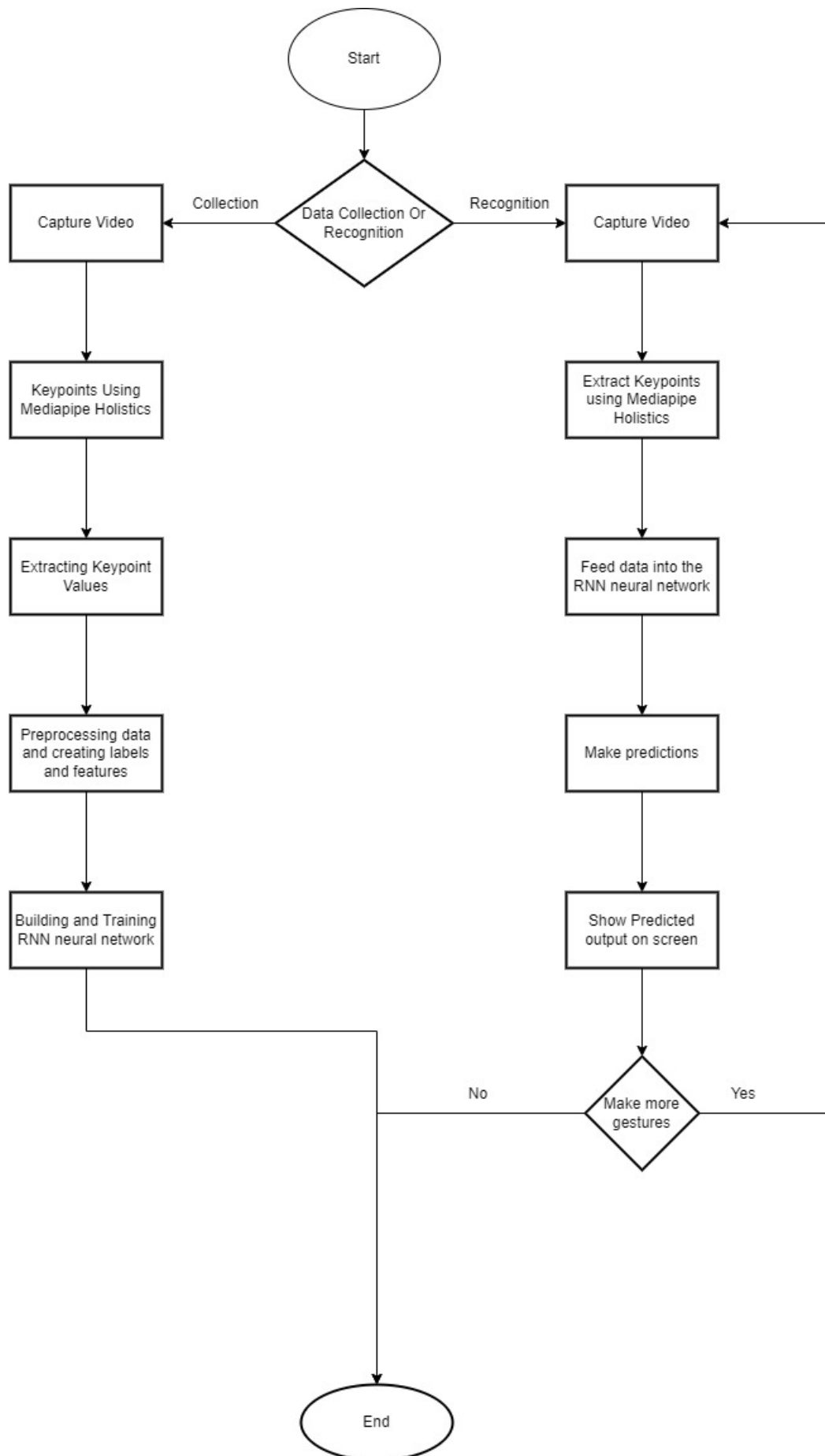


Figure 3.1: Workflow diagram

3.2 Dataset

3.2.1 Data Collection

For our data collection, we have used OpenCV combined with Mediapipe holistic. To collect our data, a webcam was used called “ Hikvision DS-U02 ”. We obtained a video resolution of 1920 X 1080 at a frame rate of 30. Mediapipe is used to obtain landmarks from different parts of the body, mainly in our model we needed the landmarks from the face, both hands, and pose. We obtain 468 landmarks from the face, 21 from each of the hands, and 33 from the pose; each of the landmarks consists of 3 coordinates x,y, and z that translate to the 3D plane, which helps the model understand the position of each of the features we are using. During collection, folders for each of the different gestures are going to be made using their corresponding words, which are categorical data. Each of the folders is going to contain 30 different samples of that specific gesture. Each of the gesture samples contains 30 frames in a sequence, and each of the frames contains all the landmarks (total of which is 1662 coordinates as $1662 = ((33 \times 4) + (21 + 21 + 468) \times 3)$). All of the coordinates are combined into an array of shapes (30,1662) and then fed into the model. For our model, we have used 15 gestures and thus a total data of 13500 (15x30x30) was collected to train our model. The used gestures and their corresponding words for our preliminary dataset are :

- Hello
- Thanks
- I Love You
- Yes
- Name
- No
- Food
- Lazy
- Time
- Me
- Give
- What
- How
- Sleepy
- I am good

After one gesture has been given, we have added a 5 second break before inserting the next gesture, this is done as we need to reset our hands, pose and facial expression before starting the next gesture, thus this gives the user enough time to start another gesture. Furthermore, after 30 gestures of the same gestures are taken, before beginning the next gesture, we have added a prompt to notify the user about what gesture is going to be taken in as input next, this thus nullifies any chance of gesture overlapping in input file due to the carelessness of the user. Here are some screenshots from our data collection phase collaged together.

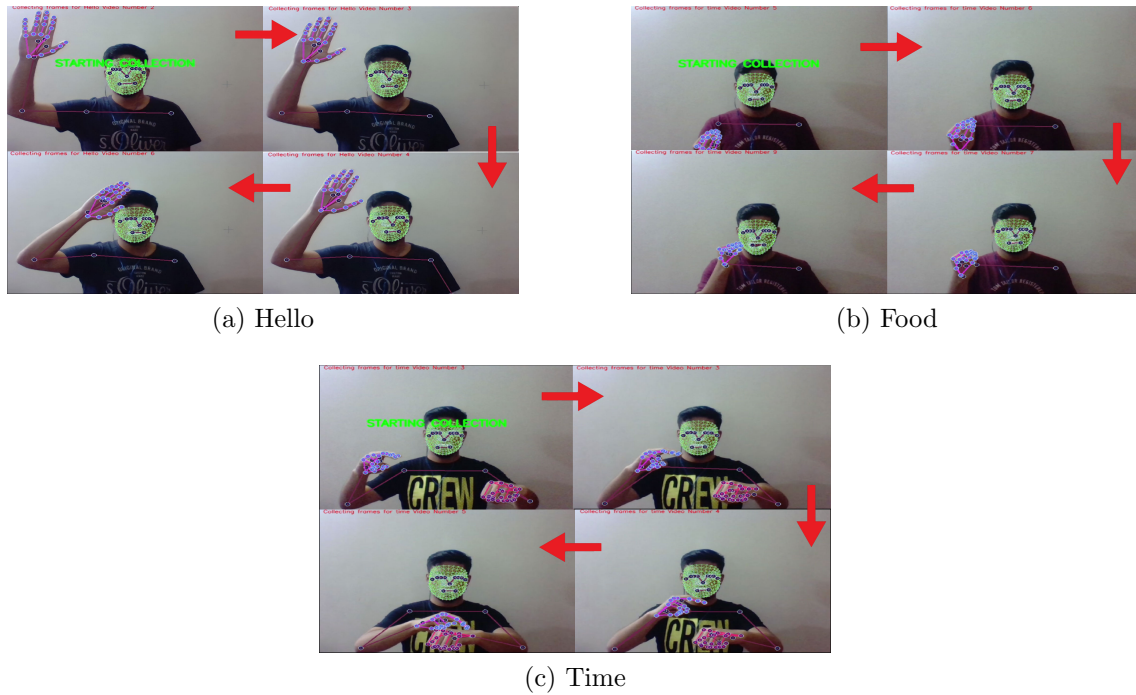


Figure 3.2: Data Collection

3.2.2 MediaPipe Holistics For Feature Extraction

MediaPipe Holistic is the pose detection library that employs not only both the hands and face but also both the feet and the entire pose of a human using landmark models. Here, for sign language recognition purposes we are required to only use the upper body of an individual, thus the parts which we are utilizing to extract features from the Holistic mode of MediaPipe are the 21 landmarks of each hand, 468 facial landmarks, and lastly the 33 body landmarks for the pose of which not all are used for the body, depending on how far we are placing our camera to portray these landmarks. Figure: 3.3 shows how the landmarks are detected.

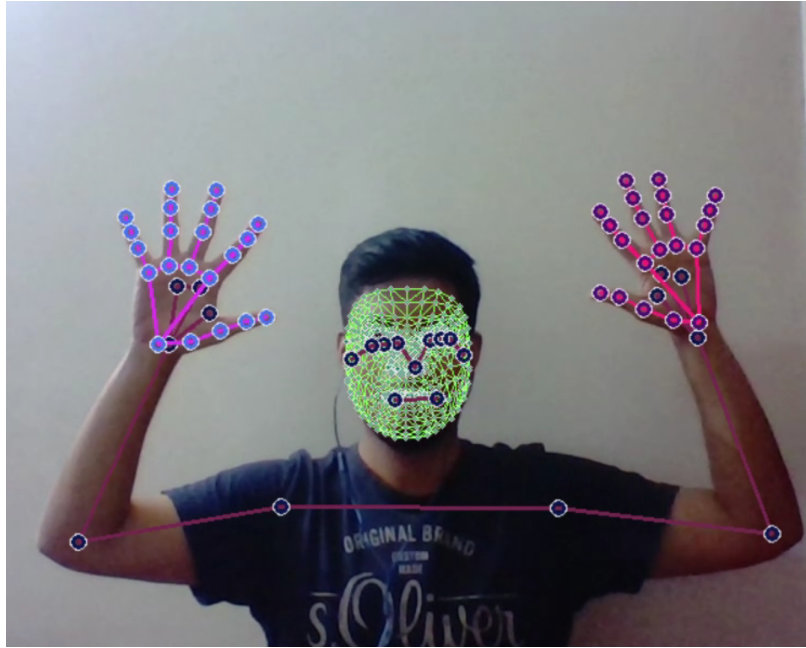


Figure 3.3: Pose Detection via MediaPipe Holistics

The landmarks for the hands, face, and pose are portrayed in real-time on the video of the action. Then the feature extraction is done by MediaPipe which translates the data points from the landmarks into coordinates of the 3D plane for each of the hands, pose and face.

3.3 Data Pre-processing

3.3.1 Resolving Null and Missing values

The collected data as discussed in 3.2.1 cannot be passed through our model as sometimes in our frame we may have missing data due to some part of the body not being present on the camera screen and thus this would cause problems when we are about to train the model. Therefore, we are going to assign “ 0 ” values to the missing points or null values to solve this issue.

3.3.2 One hot encoding

Due to the labels of our gestures (independent words for a particular gesture) being categorical data, we have decided to use one hot encoding as this makes the data easier to work with and also turns it into a readable version by the algorithm. In our case we converted the categorical data into arrays by one hot encoding. Such as: Hello=[1,0,0] , Thanks=[0,1,0] and Name=[0,0,1].

One hot encoding is beneficial for data that are not related to each other. Machine learning algorithms perceive these arrangement of numbers frequently as a means of working on them. Higher numbers are considered more significant or superior to the lower ones. Although this is useful in some ordinal scenarios, the lack of a ranking for category values in our labels might cause problems with predictions and subpar

performance, as stated [19].

For our algorithm to work, it is better to encode our data into one hot encoding for an improvement in our prediction and further processing of our data. Binary value of 0 or 1 is assigned to each categorical value, which is later used to create a new category column using one hot. Each integer value is represented by a binary vector, where the index is denoted by a 1 and the rest of the values are 0's. Simple scaling of our training data has led it to be more useful and expressive. As precise predictions are more preferable than single labels, one hot encoding is chosen, thus quickly using the probability values to match with the index.

The process of converting categorical data to one hot is shown below in the diagram 3.4. In our case, we converted the labels/words into one-hot so that it is easier to get an output from the recognition probability of all the labels later by using softmax.

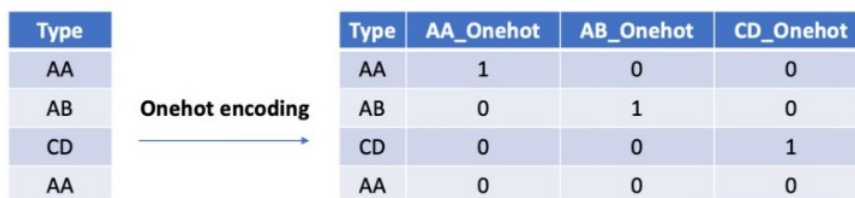


Figure 3.4: One hot encoding

3.4 System Implementation

3.4.1 Recurrent Neural Network

Recurrent Neural Network also known as RNN can be defined as one of the classes of artificial neural networks which is useful for simulating sequential data as per [28]. RNNs, which are derived from feedforward networks, behave in the same way as that of human brains. According to [15], RNN uses the same method of training data as CNN and feedforward neural networks. They stand out from other objects because of their memory, which enables them to use information from earlier inputs to influence the present input and output. In contrast to deep neural networks, which have independent outputs, RNNs depend on the previous segment of the sequence for their outputs.

However, within the RNN, there are many types available and they are as follows from the paper [18]:

- One-to-One RNN- The most common type which provides “single output for a single input”.
- One to Many RNN- This is used in many cases where it gives “multiple outputs for a single input”.
- Many to One RNN- It is used when “multiple input requires single output”.

- Many to Many RNN- This architecture gives “multiple outputs for multiple inputs” and as a result within this type they follow two different approaches and they are:
 1. $TX = TY$ where the input and output layers have the same size.
 2. $TX \neq TY$ where the input and output layers have different sizes.

Hence, as a result, we are following the Many to Many RNN model as our input is the sequence of coordinates obtained while making a gesture and our output is an array which contains the neurons activated with the percentage matched with the gestures present in the database, therefore the gesture corresponding to the highest percentage matched is displayed, using a pre-processing method.

3.4.2 LSTM and GRU for Gesture Recognition

As mentioned [22] each backpropagation will give a gradient which will further be used to update the weights required in the networks. Hence, when training any standard RNN, these will lead to the “Vanishing Gradient” problems. Thus, when dealing with longer sequences weights assigned gets smaller and smaller due to the relative gradient. As a result, a short term memory problem is occurred, where the network does not learn from any earlier inputs.

To tackle this problem speacialized versions of RNN has been created, from which two are being discussed upon

LSTMs, also known as ‘Long Short-Term Memory’ are a special type of Recurrent Neural Network with the ability to learn long-term dependencies[6]. RNNs work by connecting information found in the previous layer to the current assignment at hand. With standard RNN, problems requiring short-term memory perform great, but as soon as the program is required to remember data from a few steps back, RNN struggles. LSTM, introduced by ‘Hochreiter’ and ‘Schmidhuber’ in 1997, was designed to tackle this issue.

GRU (Gated Recurrent Unit), introduced by Cho et al. in 2014, solves the vanishing gradient problem occurring in RNN. Another variant of the LSTM, GRU gives similarly brilliant outcomes, even though they somewhat have similar construction [13].

Since each of our gestures contains 30 frames of data, RNN, with its short-term memory, fails to provide highly accurate predictions. Thus, we made the decision to use both LSTM and GRU for our model and figure out which one performs better.

3.4.3 Working Mechanism of LSTM

LSTMs have a chain-like architecture, but the repeating module is different to that of an RNN. There are four neural network layers instead of one, and they all interact differently.

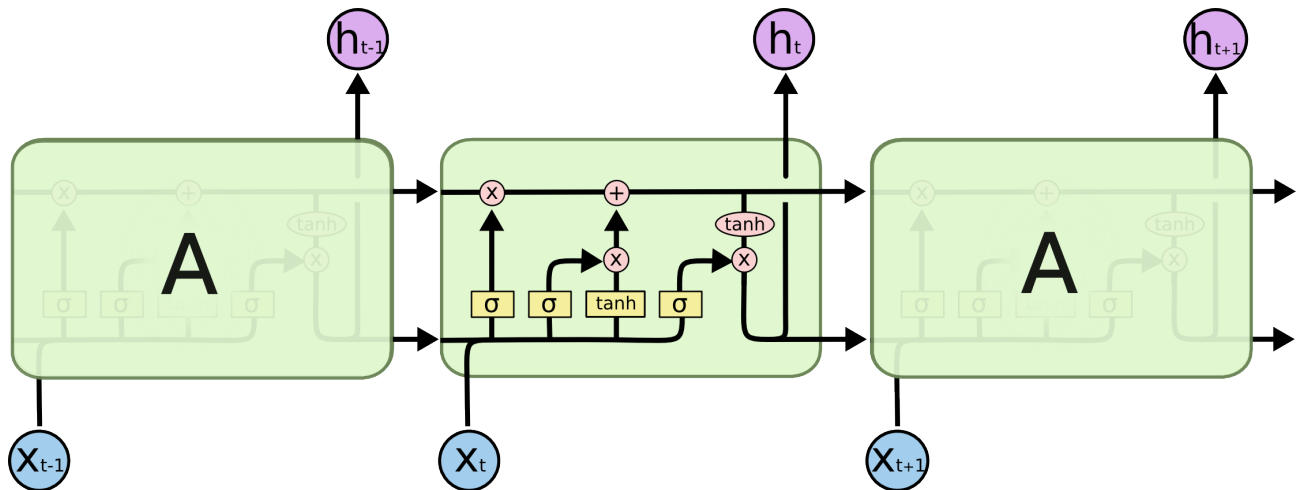


Figure 3.5: LSTM3-chain

The horizontal line running through the top of the diagram, known as the cell state, is crucial to LSTMs. It flows straight down the entire chain, carrying data along with it. There are gates that let certain data through to the cell state while rejecting others. A sigmoid neural net layer makes up these gates along with a pointwise multiplication operation. The sigmoid layer produces 0s and 1s which determine whether the information is passed onto the cell state or not. The training process is where the model learns when to open or close the gates based on the input it receives.

The initial step in the LSTM is to throw away unnecessary information and keep the relevant ones. The ‘forget gate layer’ is responsible for making this decision. It checks the input values and returns a number between 0 and 1 for each number in the cell state. A 0 output would get rid of the data while a 1 indicates that the data is important and needs to be kept.

The next step is to determine what additional data will be stored in the cell state. This has two sections. The ‘input gate layer’, determines which values to update first. A vector of potential new candidate values that could be added to the state is then produced by a *tanh* layer. These two will be combined in the next step to make a state update.

The old cell state is then updated into the new state. The forget gate layer multiplies the old state. After that, the result will be multiplied by the input gate layer and candidate values layer. This gives the new candidate value, scaled to the extent that is required to be updated for each cell state.

Finally, a decision on the parts to be outputted is made. Depending on the cell state, the output gets filtered. First, a sigmoid layer is run to determine which aspects of the cell state will be output. The cell state is then multiplied by the output of the sigmoid gate, which only outputs the desired portions, after being passed through *tanh* to compel the values to be between -1 and 1.[6] Below 3.1 to 3.6 shows equations for different parts of the LSTM unit.

$$i_t = \sigma (x_t w_{xi} + h_{t-1} w_{hi} + c_{t-1} w_{ci} + w_i \text{ bias}) \quad (3.1)$$

$$f_t = \sigma (x_t w_{xf} + h_{t-1} w_{hf} + c_{t-1} w_{cf} + w_f \text{ bias}) \quad (3.2)$$

$$z_t = \tanh (x_t w_{xz} + h_{t-1} w_{hz} + w_z \text{ bias}) \quad (3.3)$$

$$c_t = z_t \otimes i_t + c_{t-1} \otimes f_t \quad (3.4)$$

$$o_t = \sigma (x_t w_{xo} + h_{t-1} w_{ho} + c_{t-1} w_{co} + w_o \text{ bias}) \quad (3.5)$$

$$h_t = o_t \otimes \tanh (c_t) \quad (3.6)$$

,where, the input gate, forget gate, output gate, and cell gate are respectively represented by i_t , f_t , o_t , z_t . c_t and h_t are memory and output activation at time t, where t = 1, 2, 3 Tk. The Equations 3.2, 3.3, 3.5 and 3.6 are the formulas for forget, cell, output gates and hidden state [11]

3.4.4 Working mechanism of GRU

The behavior is temporal of the input data sequence that is incorporated into an RNN which is a feed-forward neural network [1]. When the sequence is lengthy, the expansion or contraction of gradients during error backpropagation may impair RNN optimization.

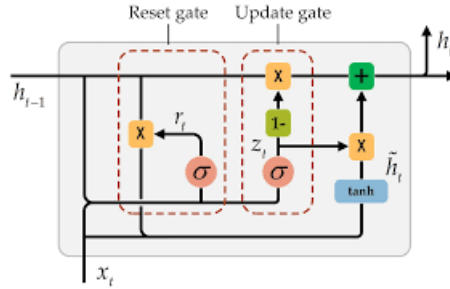


Figure 3.6: GRU cell

To solve the gradient issue, we used GRUs [4], as indicated in Figure 3.6 Update gate 'z' and reset gate 'r' are the two gates that make up a GRU. The output of the hidden layer at timestep 't' is designated as ' h_t ' in Figure 3.6, and the output vectors of these gates at time-step 't' are denoted as ' z_t ' and ' r_t ', respectively. The following is the calculation for these output vectors:

$$z_t = \sigma (W_z x_t + U_z h_{t-1} + b_z) \quad (3.7)$$

$$r_t = \sigma (W_r x_t + U_r h_{t-1} + b_r) \quad (3.8)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tanh (W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \quad (3.9)$$

where "t" is the time-step index, " σ " denotes the function sigmoid, "tanh" denotes the hyperbolic tangent function, " \circ " denotes the "Hadamard product" operation, and " b_α ," " U_α " and " W_α " denote the gate bias and weight matrix shared by the hidden units in the same layer, respectively.

3.5 Real-Time Gesture Recognition

We propose a design for a neural network that uses motion profile sequences to recognize hand gestures in real time. The network is made up of an LSTM encoder to extract the hand gestures' global temporal features and a softmax layer to calculate their conditional probabilities. The figure shows the neural network that is being constructed for our model.

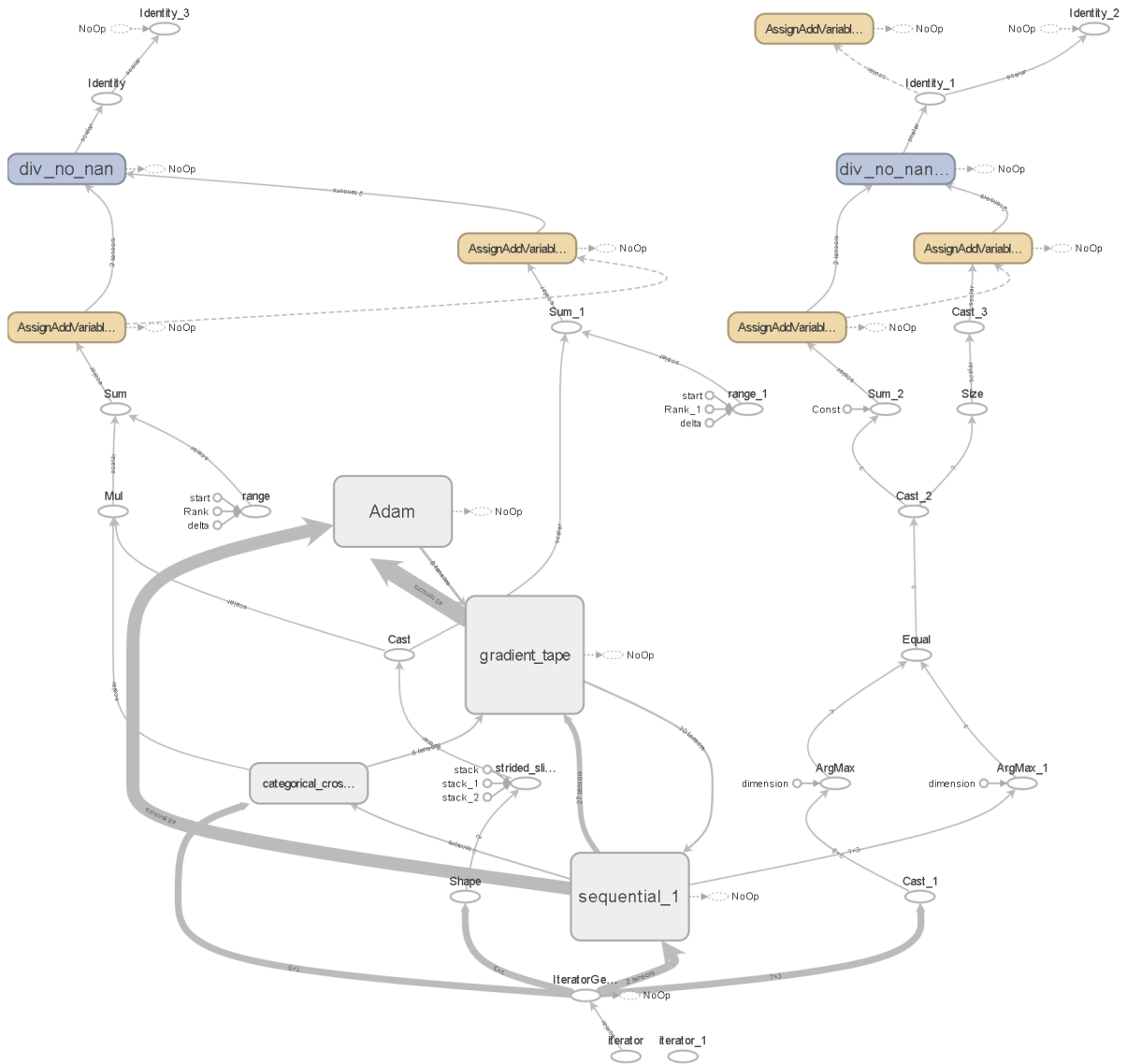


Figure 3.7: Neural Network Model

Since our model takes in landmarks saved as arrays which is needed to be fed into the input layer, these landmarks are the inputs of our model. Hence, our neural network models are of many-to-many networks, as from these inputs each neurons are to be linked to give us the final output, which are the labels of our gestures.

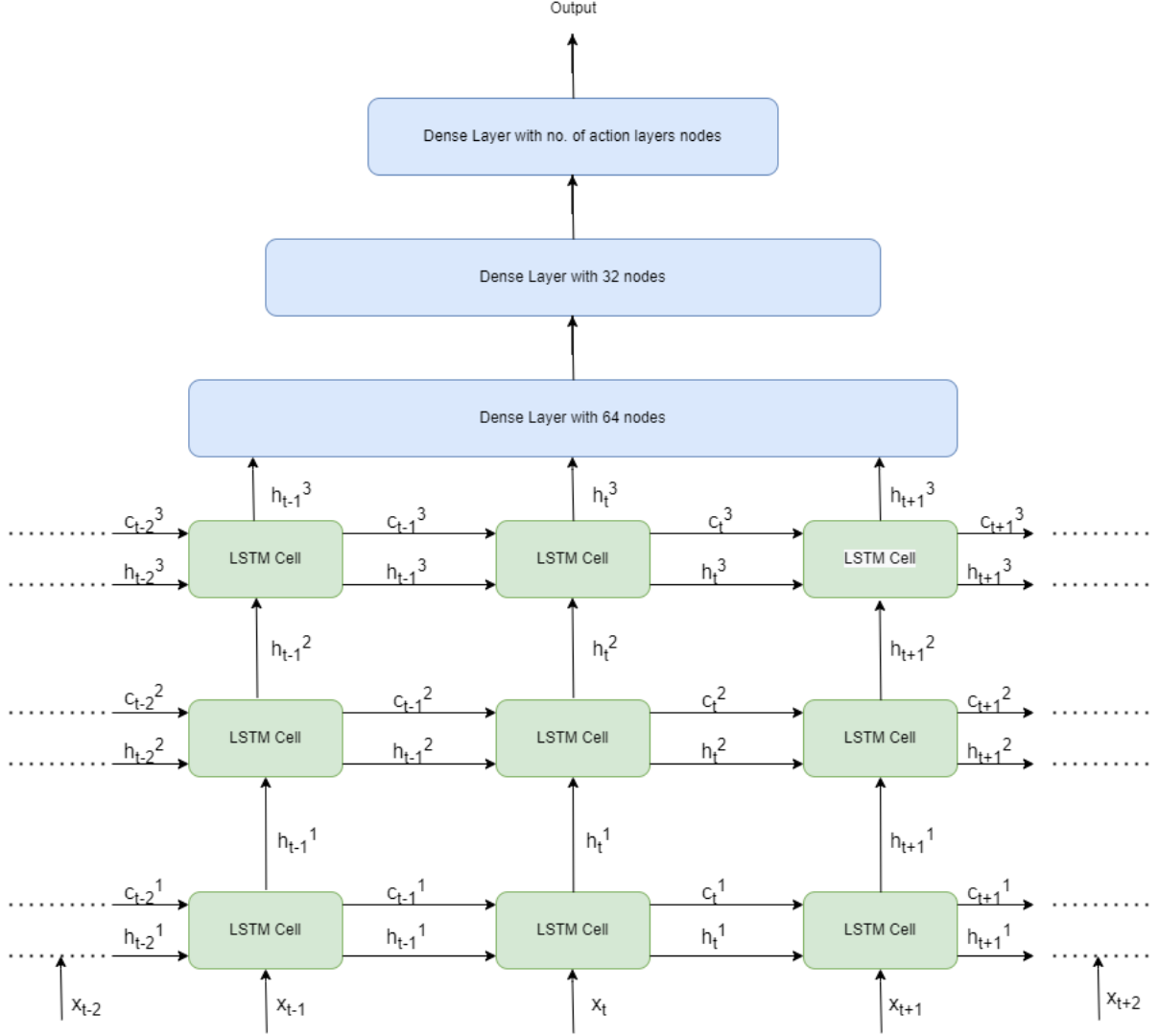


Figure 3.8: LSTM Architecture

In order to effectively extract temporal features from the 30 sequential frames extracted from an input gesture, we first use an LSTM encoder structure of 6 layers according to our architecture structure as shown 3.8. These layers are stacked together of which 3 of them are of LSTM layers and each has 64 units, 128 units and 64 units respectively, each returning the results to the next layer. The rest 3 are of the Dense Layers which works in the fully connected network layers and each has units of 64, 32 and action shape layers respectively. The gesture is then translated into an encoded vector, v . After the entire motion profile sequence has been read, the hidden state is converted to the encoded vector, $v = h_{Tk}$. The encoded vector serves as a summary of the entire sequence. In order to create the class-conditional probability, 's', the encoded vector, 'v', is connected to the softmax layer as follows, as stated [12]:

$$\mathbf{s} = \mathbf{S}(\mathbf{W}_s \mathbf{v} + \mathbf{W}_{sbais})$$

where each element, $[\mathbf{S}(\mathbf{a})]_i = e^{a_i} / \sum_j e^{a_j}$, represents the predicted probability of the i-th class.

A neuron's activation function dictates whether it should be turned on. The output

of a specific neuron is typically converted by the nonlinear functions to a value between 0 and 1, and since our model needs to decide which specific neuron it needs to activate, as a single label is needed as an output, relu activation is efficient.[15] This is represented with the formula

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

Cross entropy is used as a loss function in training. A one-hot vector with a length equal to the number of classes, C, is used to represent a ground truth label, y. The cross entropy loss between the ground truth label, y, and the predicted probability, s, can be calculated using the formula below:

$$l = - \sum_{i=1}^C y_i \log(s_i)$$

The optimizer used for this research is Adaptive Moment Estimation, also known as Adam. This optimizer is quite effective when dealing with complex issues containing a lot of data or factors. It is a combination of the “gradient descent with momentum” algorithm and the ‘RMS’ algorithm. Momentum is found as follows:

$$w_{t+1} = w_t - \alpha m_t$$

where,

$$m_t = \beta m_{t-1} + (1 - \beta) \left[\frac{\delta L}{\delta w_t} \right]$$

where,

m_t = “aggregate of gradients at time t [current]” (initially, $m_t = 0$)

m_{t-1} = “aggregate of gradients at time t-1” [previous]

w_t = “weights at time t”

α_t = “learning rate at time t”

δL = “derivative of Loss Function”

δw_t = “derivative of weights at time t”

β = “Moving average parameter (const, 0.9)”

Root Mean Square Propagation(RMSP) can be calculated using as shown in [17]:

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \epsilon)^{1/2}} \left[\frac{\delta L}{\delta w_t} \right]$$

where,

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

where,

w_t “weights at time t”

v_t = “sum of square of past gradients”. [i.e sum($\delta w_t/\delta w_{t-1}$)] (initially, $v_t = 0$)

α_t = “learning rate at time t”

δL = “derivative of Loss Function”

δw_t = “derivative of weights at time t”

$\beta =$ “Moving average parameter (const, 0.9)”

$\epsilon =$ “A small positive constant (10-8)”

Next, We again proposed another design for a neural network that will use the same motion profile sequences to recognize hand gestures in real time. This network is made up of an GRU encoder to extract the hand gestures’ global temporal features and the same softmax layer to calculate their conditional probabilities. Hence, we replaced the LSTM encoder architecture layers with that of the GRU layers and formed our sequential model. With that, we are going to later find out the differences in our model which will later be discussed in 4.3

Chapter 4

Experimental Analysis & Results

4.1 System In Operation

The model begins by capturing a video using a webcam and OpenCV as stated in 3.2.1. The most recent 30 frames are taken and the gesture is detected by comparison of the sequential images. Mediapipe takes each of these frames and assigns landmarks to them. The face, hands and body get differently labeled keypoints. These are then fed into our both LSTM encoder and GRU encoder, where the data is converted to a vector that can be used to generate a probability value. The value found is compared with existing data present within the dataset to try and find the closest match. Accuracy values are assigned to all available gestures in the library. It is also worth mentioning that as we are using softmax, all the words in the dataset obtain an individual percentage of accuracy, all of this adds up to 100%. Since, our model has a threshold accuracy score of 70%, thus the gesture that has accuracy higher than the threshold is selected as the model's predicted answer. This is displayed on the screen and the model waits for more input from the user. Additionally, our model disregards any word in case it appears more than once consecutively. This makes sure that if more than 30 frames are assigned to a gesture, the system does not display redundant data. Furthermore, using the OpenCv feed we are portraying the output where the sequence of respective word's gestures translate into complete sentences.

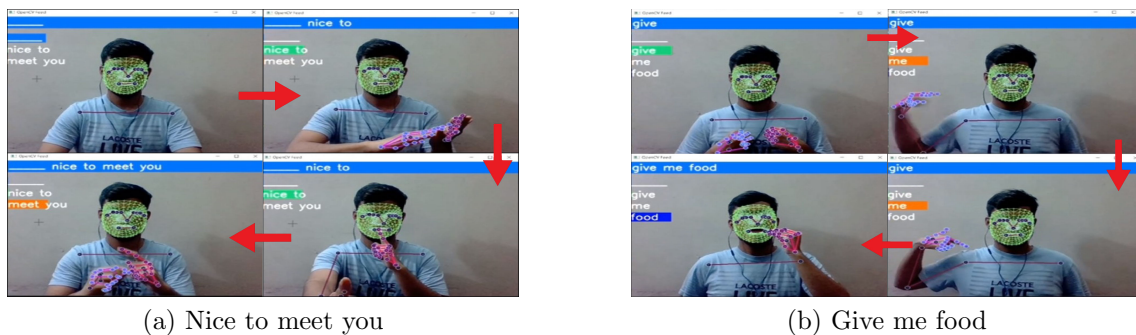


Figure 4.1: Predicted Output

And this shows the individual words being predicted when a gesture is made and the outputs are shown on the screen.

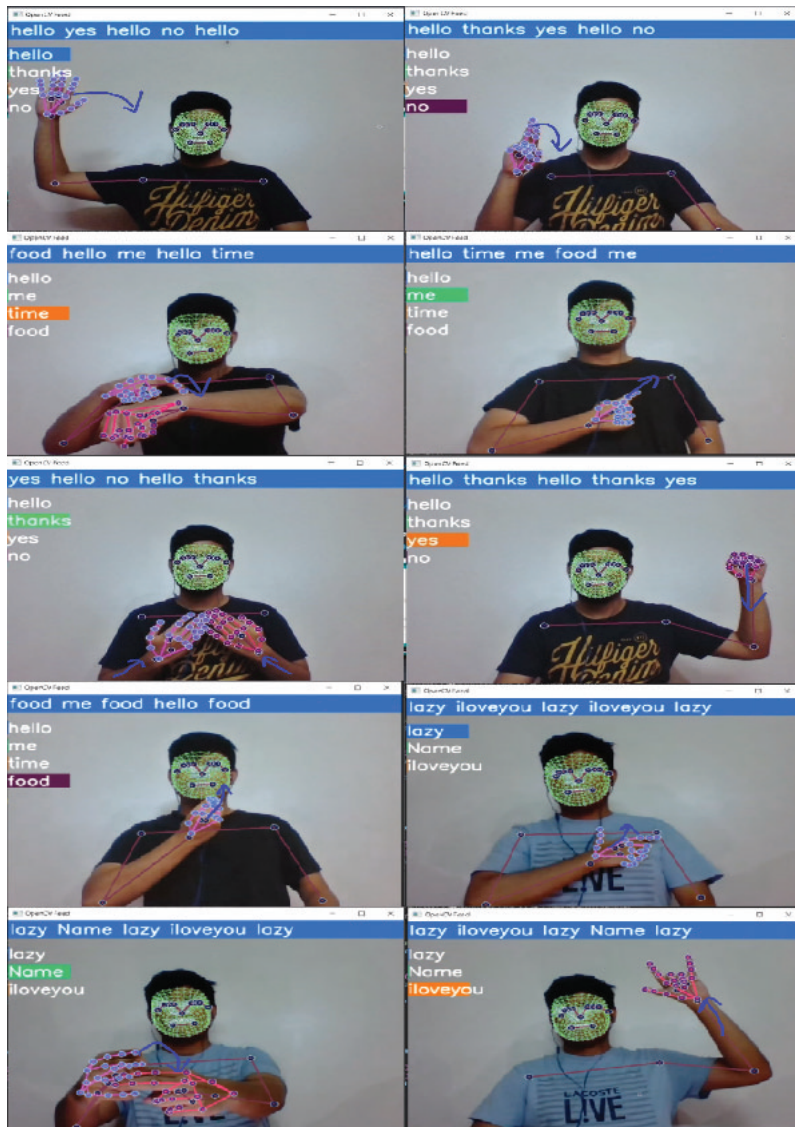


Figure 4.2: Results

4.2 Training and Testing Sets

In our model, we obtained 15 gesture sets each of which was repeated 30 times in a different way. Furthermore, each gesture contains 30 frames, thus leading to us having a dataset of 13500 frames. The 543 landmarks (containing 1662 coordinates) of the frames are collected into an array of shapes (450,30,1662). We then split this data in the ratio of 90:10 by a process of random selection. This ratio is chosen as our model benefits from being trained with larger sets of data as opposed to smaller ones. Therefore using larger sets of data improves the accuracy of our model. Finally, manual testing is done by a group of 5 different users. When a user makes a gesture, our model takes the most recent 30 frames and uses them to predict the gesture label. After it matches with a label from our trained set, the matched percentage is shown and also the word is printed on the screen. Further gestures given as input would be processed by the model and words would be displayed as before thus forming a full-fledged sentence.

Now, when we run both of our models, the parameters of our neural network are shown. These normally are the connection weights, which in this instance are picked up throughout the training phase. Thus, the algorithms themselves (and the input data) tune these parameters.

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 30, 64)	442112
lstm_4 (LSTM)	(None, 30, 128)	98816
lstm_5 (LSTM)	(None, 64)	49408
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 32)	2080
dense_5 (Dense)	(None, 3)	99

=====
 Total params: 596,675
 Trainable params: 596,675
 Non-trainable params: 0

(a) LSTM parameters

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 30, 64)	331776
gru_1 (GRU)	(None, 30, 128)	74496
gru_2 (GRU)	(None, 64)	37248
dense_6 (Dense)	(None, 64)	4160
dense_7 (Dense)	(None, 32)	2080
dense_8 (Dense)	(None, 3)	99

=====
 Total params: 449,859
 Trainable params: 449,859
 Non-trainable params: 0

(b) GRU parameters

Figure 4.3: Model Summary

Typically, the hyper parameters are the “learning rate”, “batch size”, and “number of epochs”. Both LSTM and GRU were found to use less training parameters compared to other neural networks, such as CNN. Thus, when using our model, it was found that GRU uses 449,859 training parameters, whereas LSTM uses 596,675 training parameters, and both uses 0 non-training parameters. This makes our training model run more efficiently and smoothly.

4.3 Results

Our main objective with the suggested model is to detect American Sign Language properly and reliably in the shortest amount of time. Because we created our model for the benefit of those who are challenged with hearing and speech, our model should be able to acknowledge gestures not only precisely but also at a fast enough rate that will allow communication in real-time. As per the artificial neural network’s context, an “epoch” is one loop of the whole training dataset. Usually, it takes more than a few “epochs” to train a neural network. Since both GRU and LSTM have the architecture necessary to provide solutions for our model, we tested each of them to determine which neural network performs better. While train-testing our model consisting of a minimum of 5 words, we found both GRU and LSTM reach an accuracy of 100% by observing the accuracy against epoch graphs. However, GRU, shown in figure 4.4, reaches that peak accuracy in a lot less epochs compared to LSTM, figure 4.5.

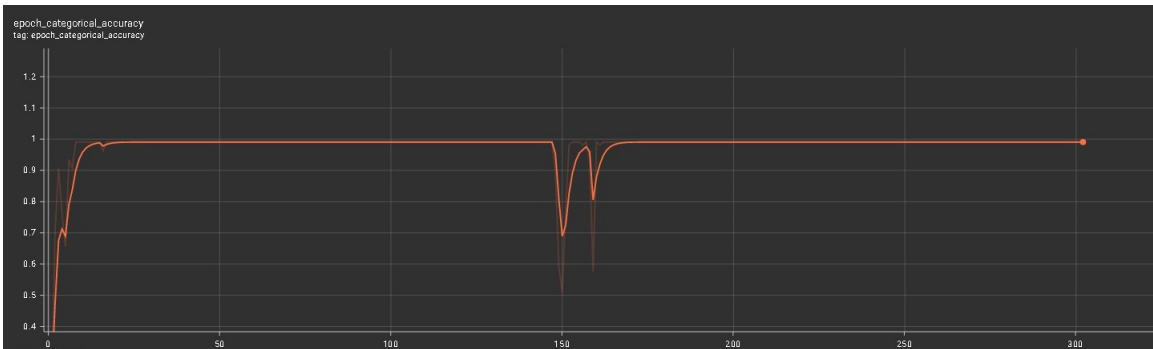


Figure 4.4: GRU accuracy epoch graph



Figure 4.5: LSTM accuracy epoch graph

Judging by the numbers displayed by these graphs, and the use of parameters as defined in 4.2 GRU might seem to be the obvious choice for our research as it is significantly faster than LSTM. However, according to [29] GRU’s performance in smaller datasets is fantastic, but as soon as the dataset starts getting bigger, its accuracy starts dropping off and LSTM outperforms it. Since these tests were carried out on a small library containing five words, the results came in GRU’s favour.

When more words are added to the library, LSTM would become the ideal neural network for this model. This problem occurs due to the major difference in LSTM and GRU, where one of the gates in LSTM, known as the Output Gate, is not present in GRU. Thus, GRU does not possess any internal memory. This memory is solely the reason LSTM can be more accurate on a larger dataset, as stated [22]

So, keeping that in mind, after running both our models, we have decided to keep our LSTM model as the final decision when implementing our larger datasets.

We generated a confusion matrix for two words from our database. The train-test split was in the ratio of 90 to 10. The confusion matrix shown below, figure 4.7 and 4.6, was made using the test data. Both the words returned high accuracy scores with only one false data obtained in each case. This shows the reliability of our models.

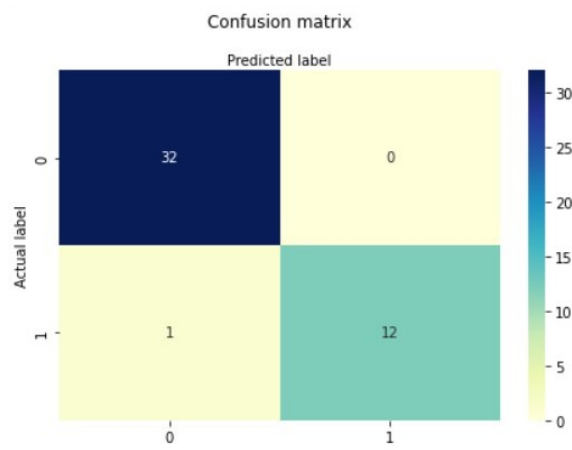


Figure 4.6: Hello

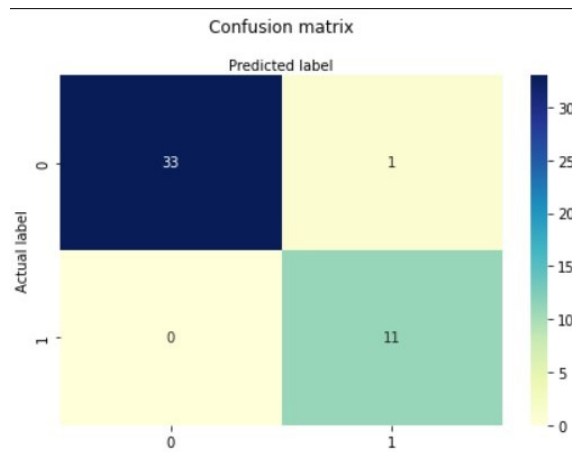


Figure 4.7: Food

Chapter 5

Conclusion And Future Scope

5.1 Conclusion

To conclude, we can say that sign language recognition is a necessity for deaf and mute people to communicate. According to WHO, there are around 5% of hearing and speech impaired people around the world. Through this research, our main aim was to find a way to remove the communication barrier between deaf-mutes and normal people. To us, solving this issue seemed like a great opportunity to build a real-time sign language model with the help of MediaPipe Holistic and Frame Composition LSTM. Hence from this model, we have received an accuracy of 98.8% and it stands out from the existing models because it recognizes the motion in real-time whereas some only work with still or static hand gesture images or relies on complex detection system hardware such as gloves or radar sensors.

5.2 Future Scope

In the future, we will be working on our current model and making it more efficient by expanding the quantity of our dataset. Despite the increment in the amount of data we also have a plan to implement Natural Language Processing (NLP) in our dataset which will not only fix the grammatical issue as ASL has different grammar than Basic “English”. Other than this, our model is not only restricted to ASL it can be used to train and test any different sign language like CSL(Chinese Sign Language), BdSL (Bangladesh Sign Language), and many others. As a result, this diversity is only possible due to our implementation of MediaPipe Holistic and Frame Composition LSTM. Our framework not only allows hand gesture detection but amongst a wide range of applications it can also be used for Attention Span Tracking in a Classroom or office, Fire Detection by tracking the motion of individuals or multiple people, and lastly even detecting if an individual is doing YOGA poses correctly. Also, we are trying to create an application that can be used on our smartphones. Due to its real-time accuracy measurement, our model is already more effective than most previous models. The system was created exclusively for research purposes, although it may also be used for commercial purposes. Nevertheless, helping those who are deaf or hard of hearing continues to be our first priority.

References

- [1] L. R. Jain L. C. & Medsker, *Recurrent Neural Networks: Design and Applications*, vol. 1st ed. 1999.
- [2] *How many deaf people are there in the United States? Estimates from the Survey of Income and Program Participation - PubMed*, <https://pubmed.ncbi.nlm.nih.gov/16177267/>, [Online; accessed 2022-09-20], Sep. 2005.
- [3] R. E. Mitchell, T. A. Young, B. Bachleda, and M. A. Karchmer, “How Many People Use ASL in the United States? Why Estimates Need Updating,” *Sign Language Studies*, vol. 6, no. 3, pp. 306–335, 2006.
- [4] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, 2014. DOI: arXiv:1406.1078.
- [5] H. D. Yang, “Sign Language Recognition with the Kinect Sensor Based on Conditional Random Fields,” *Sensors*, vol. 15, no. 1, pp. 135–147, Dec. 2014.
- [6] C. Olah, “Understanding LSTM Networks – colah’s blog,” *Understanding LSTM Networks – colah’s blog*, Aug. 2015.
- [7] *3 Main Types of Communication — A-State Online*, <https://degree.astate.edu/articles/undergraduate-studies/3-main-types-of-communication.aspx>, [Online; accessed 2022-09-20], Sep. 2016.
- [8] *An analysis of Convolutional Long Short-Term Memory Recurrent Neural Networks for gesture recognition - ScienceDirect*, <https://www.sciencedirect.com/science/article/pii/S0925231217307555>, [Online; accessed 2022-09-21], May 2017.
- [9] S. F. Hasan, T. M. Rahman, A. R. Chowdhury, and A. Biswas, *Gesture based implementation on neural network: Bengali sign language to text conversion*, <http://hdl.handle.net/10361/9064>, [Online; accessed 2022-09-21], Jan. 2017.
- [10] C. Li, Z. Peng, T. Y. Huang, *et al.*, “A Review on Recent Progress of Portable Short-Range Noncontact Microwave Radar Systems,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 5, pp. 1692–1706, May 2017.
- [11] N. L. Hakim, T. K. Shih, S. P. Kasthuri Arachchi, W. Aditya, Y. C. Chen, and C. Y. Lin, “Dynamic Hand Gesture Recognition Using 3dcnn and LSTM with FSM Context-Aware Model,” *Sensors*, vol. 19, no. 24, p. 5429, Dec. 2019.
- [12] C. JAE-WOO, R. SI-JUNG, and K. JONG-HWAN, *Short-Range Radar Based Real-Time Hand Gesture Recognition Using LSTM Encoder*, <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8662554>, [Online; accessed 2022-09-21], Mar. 2019.

- [13] S. Kostadinov, *Understanding GRU Networks. In this article, I will try to give a...* — by Simeon Kostadinov — *Towards Data Science*, <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>, [Online; accessed 2022-09-25], Nov. 2019.
- [14] N. T. Do, S. H. Kim, H. J. Yang, and G. S. Lee, *Applied Sciences — Free Full-Text — Robust Hand Shape Features for Dynamic Hand Gesture Recognition Using Multi-Level Feature LSTM*, <https://www.mdpi.com/2076-3417/10/18/6293>, [Online; accessed 2022-09-21], Sep. 2020.
- [15] I. C. Education, *What are Recurrent Neural Networks?* <https://www.ibm.com/cloud/learn/recurrent-neural-networks>, [Online; accessed 2022-09-25], Sep. 2020.
- [16] P. S. Neethu, R. Suguna, and D. Sathish, “An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks,” *Soft Computing*, vol. 24, no. 20, pp. 15 239–15 248, Mar. 2020.
- [17] R. Prakhar, *Intuition of Adam Optimizer - GeeksforGeeks*, <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>, [Online; accessed 2022-09-21], Oct. 2020.
- [18] Y. Prasad, *Types of RNN (Recurrent Neural Network)*, <https://iq.opengenius.org/types-of-rnn/>, [Online; accessed 2022-09-25], Feb. 2020.
- [19] A. Thakur, *How One-Hot Encoding Improves Machine Learning Performance*, <https://wandb.ai/ayush-thakur/dl-question-bank/reports/How-One-Hot-Encoding-Improves-Machine-Learning-Performance-VmllzoxOTkzMDk>, [Online; accessed 2022-09-21], Aug. 2020.
- [20] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, *Remote Sensing — Free Full-Text — Hand Gestures Recognition Using Radar Sensors for Human-Computer-Interaction: A Review*, <https://www.mdpi.com/2072-4292/13/3/527>, [Online; accessed 2022-09-21], Feb. 2021.
- [21] *Camera, Radar and LiDAR: A Comparison of the Three Types of Sensors and Their Limitations - AUTOCRYPT %*, <https://autocrypt.io/camera-radar-lidar-comparison-three-types-of-sensors/>, [Online; accessed 2022-09-21], Aug. 2021.
- [22] V. Lendave, *Lstm Vs GRU in Recurrent Neural Network: A Comparative Study*, <https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/>, [Online; accessed 2022-09-25], Aug. 2021.
- [23] A. Mujahid, M. J. Awan, A. Yasin, *et al.*, *Applied Sciences — Free Full-Text — Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model*, <https://www.mdpi.com/2076-3417/11/9/4164>, [Online; accessed 2022-09-20], May 2021.
- [24] *Python - Facial and hand recognition using MediaPipe Holistic - GeeksforGeeks*, <https://www.geeksforgeeks.org/python-facial-and-hand-recognition-using-mediapipe-holistic>, [Online; accessed 2022-09-21], Sep. 2021.
- [25] *What Is American Sign Language (ASL)? — NIDCD*, <https://www.nidcd.nih.gov/health/american-sign-language>, [Online; accessed 2022-09-20], Oct. 2021.
- [26] G. Boesch, *A Guide to OpenPose in 2022 - viso.ai*, <https://viso.ai/deep-learning/openpose/>, [Online; accessed 2022-09-21], Jun. 2022.

- [27] , *Detection of human body landmarks - MediaPipe and OpenPose comparison* — *HearAI*, <https://www.hearai.pl/post/14-openpose/>, [Online; accessed 2022-09-21], Apr. 2022.
- [28] N. Donges, *Recurrent Neural Networks (RNN): What It Is & How It Works* — *Built In*, <https://builtin.com/data-science/recurrent-neural-networks-and-lstm>, [Online; accessed 2022-09-25], Jul. 2022.
- [29] *Comparison of GRU and LSTM in keras with an example -*, <https://www.projectpro.io/recipes/what-is-difference-between-gru-and-lstm-explain-with-example>, [Online; accessed 2022-09-25].
- [30] J. Lapiak, *Sign Language • ASL* — *HandSpeak®*, <https://www.handspeak.com/>, [Online; accessed 2022-09-20].
- [31] *Sign Language MNIST*, <https://www.kaggle.com/datasets/datamunge/sign-language-mnist>, [Online; accessed 2022-09-21].
- [32] *Sign language recognition using sensor gloves*, <https://ieeexplore.ieee.org/document/1201884>, [Online; accessed 2022-09-21].