

**SIGN LANGUAGE RECOGNITION AND TRANSLATION USING DEEP  
LEARNING**

**A PROJECT REPORT**

*Submitted by*

**ABINANDHAN A**

**ARAVIND A**

**SURIYAA M M**

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF ENGINEERING**

*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**



**PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH,  
COIMBATORE - 641 062**

**ANNA UNIVERSITY:: CHENNAI 600 025**

**APRIL 2021**

**ANNA UNIVERSITY : CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **SIGN LANGUAGE RECOGNITION AND TRANSLATION USING DEEP LEARNING** is the bonafide work of **ABINANDHAN A (715517106001), ARAVIND A (715517106303), and SURIYAA M.M. (715517106053)** who carried out the project work under my supervision.

**SIGNATURE**

Dr. P. Vetrivelan MS., Ph.D.,  
**HEAD OF THE DEPARTMENT**  
Associate Professor  
ECE Department  
PSG Institute of Technology And  
Applied Research  
Coimbatore - 641 062

**SIGNATURE**

Dr. P. Vetrivelan MS., Ph.D.,  
**HEAD OF THE DEPARTMENT**  
Associate Professor  
ECE Department  
PSG Institute of Technology And  
Applied Research  
Coimbatore - 641 062

Submitted for the Anna University Viva-Voce examination held on

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We would like to thank the **Management** of PSG Institute of Technology and Applied Research for providing us with excellent facilities for the completion of this project.

We are grateful to **Dr. G. Chandramohan**, Principal of PSG Institute of Technology and Applied Research for giving us support throughout the project.

We are thankful to our guide **Dr. P. Vetrivelan**, Professor and Head, Department of Electronics and Communication Engineering, for his constant support throughout the project and helping us complete the project successfully.

We also would like to thank all the faculty members and staff of the Electronics and Communication Engineering Department for their kind cooperation and encouragement during the course of this work.

## **ABSTRACT**

Sign Languages are used for communication by the people with hearing and speaking disabilities. Even though these languages are common within the community, many people cannot interpret these signs. This creates a “language barrier” thus inhibiting communication among people with hearing and speaking disabilities and the others. It is difficult for people to learn all the different sign languages, since there are many versions. The seemingly big learning curve and probable infrequent encounters where sign language is used for communication poses a big challenge for the people with hearing and speaking disabilities to be understood. This is where a translation application from sign language is useful. There is significant ongoing research that aims to create models based on deep learning which are best in class for gesture recognition. The dataset for the hand signs has been expanding ever since the research began and more and more accurate models are being developed.

The implementation of a translation application is the main purpose of this project. Microcontrollers and cameras are used to capture videos and process them using convolutional and recurrent neural networks. A small dataset was created specifically for training purposes in this project to counter the resource constraints. The convolutional and recurrent parts of the network extract the spatial and temporal features within the video, respectively. They finally predict the class of the captured video frames, stored in a buffer until being passed to the network. The predictions from the neural network, mapped to the classes, are then converted into audio signals which are output by a speaker attached to the microcontroller. Thus the entire pipeline allows for the people to interpret and understand the sign language without requiring to learn the sign language itself.

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>4</b>
	<b>LIST OF TABLE</b>	<b>10</b>
	<b>LIST OF FIGURES</b>	<b>11</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>14</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>15</b>
1.1	NEED FOR THE PROJECT	15
1.2	SIGN LANGUAGE GESTURE TRANSLATION SYSTEM	16
1.3	SOLUTION APPROACH	16
1.3.1	Video Frame Collection	16
1.3.2	Passing the frames through the Neural Network	17
1.3.2.1	Convolutional Neural Networks	17
1.3.2.2	3D-CNN	18

1.3.2.3	Long Short-Term Memory	18
1.3.2.4	ConvLSTM	18
1.3.3	Class Prediction and Audio Output	19
1.4	ORGANIZATION OF CHAPTER	19
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>20</b>
<b>3</b>	<b>COMPUTER VISION</b>	<b>23</b>
3.1	INTRODUCTION	23
3.1.1	Applications	23
3.2	IMAGE PROCESSING	24
3.3	FEATURES	25
3.4	FEATURE DETECTION	26
3.5	FEATURE EXTRACTION	26
3.6	DATASET	27
<b>4</b>	<b>HARDWARE DESCRIPTION</b>	<b>30</b>
4.1	RASPBERRY PI	30
4.1.1	Hardware	31
4.1.1.1	Model B+ Specifications	34
4.1.2	Need for Raspberry Pi	34

4.1.3	Setting up Raspberry Pi	35
4.2	CAMERA	35
4.2.1	System Requirements	36
4.2.2	Features	36
4.3	SPEAKER	36
4.3.1	Specifications	37
<b>5</b>	<b>DEEP LEARNING</b>	<b>38</b>
5.1	EVOLUTION OF DEEP LEARNING	38
5.1.1	Neural Network	39
5.2	CONCEPTS OF DEEP LEARNING NEURAL NETWORK	40
5.3	CONVOLUTIONAL NEURAL NETWORK (CONVNET)	43
5.4	LAYERS OF CNN	44
5.4.1	Convolutional Layer	44
5.4.2	Pooling Layer	46
5.4.3	Fully Connected Layer	47
5.5	CNN ARCHITECTURE	48
5.6	RECURRENT NEURAL NETWORK	49

5.6.1	Vanishing Gradient Problem	51
5.6.2	Exploding Gradient Problem	52
5.7	LONG SHORT-TERM MEMORY	52
5.8	CONVLSTM LAYER	55
<b>6</b>	<b>PROPOSED METHODOLOGY</b>	<b>57</b>
6.1	CAMERA	58
6.2	PROPOSED DEEP LEARNING ALGORITHM	58
6.2.1	ConvLSTM Layer	59
6.2.2	Dense Layer	59
6.3	AUDIO OUTPUT	61
<b>7</b>	<b>RESULT ANALYSIS</b>	<b>62</b>
7.1	RESULTS FROM THE COLLECTION OF DATASET	62
7.2	RESULTS OBTAINED FROM TRAINING OF THE NEURAL NETWORK	64
7.3	RESULTS FROM THE PREDICTIONS USING THE MODEL FOR IMPLEMENTATION	66



<b>8</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>68</b>
8.1	CONCLUSION	68
8.2	FUTURE WORK	68
	<b>REFERENCES</b>	<b>70</b>
	<b>APPENDIX</b>	<b>72</b>

## LIST OF TABLES

<b>TABLE NO.</b>	<b>TABLE NAME</b>	<b>PAGE NO.</b>
4.1	Comparison of model A, model B, model B+	30
7.1	Organization of the classes representing words	64

## LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	Representation of labelled data	28
4.1	Raspberry Pi model B+ and its pin configuration	31
4.2	Logitech webcam - C170	36
4.3	Speaker module	37
5.1	A basic Neural Network structure	39
5.2	Modelling of Artificial Neurons	42
5.3	An example of a CNN architecture	44
5.4	Example of ReLU function	46
5.5	Example for Fully Connected Layer	47
5.6	Recurrent Neural Network architecture	49
5.7	Composition of an RNN from a dense neural network	50
5.8 (a)	One to One	50
5.8 (b)	One to Many	50
5.8 (c)	Many to One	51

5.8 (d)	Many to Many	51
5.9	Represents the possibility of losing information due to vanishing gradient	52
5.10	Long Short-Term Memory	53
5.11 (a)	Forget gate decision	55
5.11 (b)	Input gate decision	55
5.11 (c)	Updating the old cell state into the new cell state	55
5.11 (d)	Output gate decision	55
5.12	Structure of a ConvLSTM network	56
6.1	Block diagram of the recognition and translation process	57
6.2	The proposed neural network architecture	59
6.3	An example of the softmax function	60
7.1	Organization of the dataset into train and test sets	62
7.2	Organization of the classes to be trained	62
7.3	Collection of videos of a class	63
7.4	A 640 x 480 frame	63

7.5	Neural network model along with the number of parameters	65
7.6	Results after each epoch of the training process	65
7.7 (a)	Accuracy vs Number of epochs	66
7.7 (b)	Loss vs Number of epochs	66
7.8 (a)	Prediction for “Hello”	67
7.8 (b)	Prediction for “Yes”	67
7.8 (c)	Prediction for “Friend”	67

## LIST OF ABBREVIATIONS

<b>1D</b>	One Dimension
<b>2D</b>	Two Dimension
<b>3D</b>	Three Dimension
<b>CNN</b>	Convolutional Neural Network
<b>LSTM</b>	Long Short-Term Memory
<b>TGCN</b>	Temporal Graph Convolutional Network
<b>RELU</b>	Rectified Linear Unit
<b>RGB</b>	Red, Green, Blue
<b>ASL</b>	American Sign Language
<b>CV</b>	Computer Vision
<b>CPU</b>	Central Processing Unit
<b>GPU</b>	Graphics Processing Unit
<b>GPIO</b>	General Purpose Input/ Output
<b>RAM</b>	Random Access Memory
<b>UART</b>	Universal Asynchronous Receiver Transmitter
<b>PC</b>	Personal Computer
<b>SD</b>	Secure Digital
<b>USB</b>	Universal Serial Bus
<b>ML</b>	Machine Learning
<b>ANN</b>	Artificial Neural Network

## **CHAPTER 1**

### **INTRODUCTION**

Sign languages are languages that convey meaning using visual cues with the help of hand gestures and expressions. These languages have specific grammar and lexicon that go along with them. They are not universal but have similarities among them. Thus sign languages are like any other language and have to be learnt. These languages are primarily used by people who are unable to speak or hear. Most countries use a specific sign language such as ASL (American Sign Language) in the USA and Canada, French Sign Language used in France and British, Australian and New Zealand Sign Language used in their respective countries. International Sign, formerly referred as Gestuno, is the main language used in events involving people with hearing disabilities.

#### **1.1 NEED FOR THE PROJECT**

Due to infrequent communication using the sign language most people encounter, it is unlikely that they would be willing to undergo the learning curve involved. It thus creates situations where the people with hearing or speaking difficulties are unable to express themselves even if they have a language that they can communicate with. Also, the diversity makes it very hard to know all the signs from the various languages. Thus, a translation application that is able to detect the hand signs and convert it into the required audio signals to be intelligible to the sign language unschooled. A portable device that can be carried by the people can avoid the situations where the people are unable to communicate due to sign language barrier. They can be made compatible with various versions of the sign language and the user does not require to undergo any learning. It thus enables

people with hearing and speaking disorders to communicate without any expectations from the listener.

## **1.2 SIGN LANGUAGE GESTURE TRANSLATION SYSTEM**

Gesture Translation System aims at capturing the features and their movement involved in a video having gestures, classifying them, and outputting them as audio signals. It enables the identification of hand signs made by the users of the Sign Language and return the word they represent. It helps in alleviating the inconvenient situations that can occur if people involved in communication use Sign Language and the listener does not know how to interpret it. It makes it easier for any Sign Language to be used, thereby avoiding the need to know various signs for the same word representation. Gesture recognition is done with the help of models based on deep learning and it is processed further to output audio.

## **1.3 SOLUTION APPROACH**

The gesture translation system is implemented by using a camera, microcontroller, speaker and learning algorithms. Various libraries like Tensorflow and PyAudio are used in the process.

The pipeline for sign recognition is as follows,

1. Video frame collection in buffer
2. Passing the frames in the buffer through the neural network
3. The class is predicted and outputted in the form of sound signals

### **1.3.1. VIDEO FRAME COLLECTION**

Camera connected to the microcontroller captures the video feed frame by frame and stores the individual frames in a buffer. The video is resized to the required format so as to be compatible to be passed onto the next layer.



### **1.3.2 PASSING THE FRAMES THROUGH THE NEURAL NETWORK**

The neural networks form the basis of the translation system as they perform the classification of videos. In contrast to the 2D Convolutional Neural Networks that can identify images based on spatial features, a model that is capable of extracting spatial as well as temporal features is required to accommodate the video data, essentially a collection of images. As the collection of images from a 3D array, 3D CNNs can be used. 2D Convolution and LSTM can be used in conjunction to perform the same operation. Various other networks like Pose-based TGCN and Multilayer Perceptrons can be used. The 2D Convolution + LSTM model provided more accurate results for the dataset created.

#### **1.3.2.1 CONVOLUTIONAL NEURAL NETWORKS**

Gesture recognition involves the classification of videos to their appropriate word classes. A convolutional neural network is a type of model architecture used in deep learning algorithms. The CNN model is similar to the collection of neurons present in the brains. The complete convolutional neural network is divided into feature learning and image classification. This deep learning algorithm contains input, output and hidden layers. Feature learning consists of the input layer, the convolutional layer, ReLU (activation function), and pooling layers. Image classification is done with the help of flattened, fully connected, softmax layers. These hidden layers undergo a convolutional operation. The inputs and outputs of these layers are passed to the activation functions and lastly convolution and therefore these layers are called as hidden layers. Since the regular 2D CNNs are not capable of detecting the temporal features, i.e., the correlated features between the frames, 3D CNNs can be used.

### **1.3.2.2 3D-CNN**

In a 3D-CNN, the frames are taken as 3-dimensional arrays and then passed onto the layers with convolution operations with 3D filters that capture spatial and temporal features, thus ensuring the connection between frames is learned by the model. It helps in capturing motion information that is crucial for identifying the hand gesture. In a convolution operation on 3-dimensions, a filter is able to traverse in 3 directions namely the width, height and channels of the respective image. At every pixel, addition and matrix element-wise multiplication output a single number. The output from the 3-dimensional convolutional operations are a 3D matrix, due to the fact that the filter traverses through a 3-dimensional domain.

### **1.3.2.3 LONG SHORT-TERM MEMORY**

Long short-term memory (LSTM) is an RNN (Recurrent Neural Networks) model based architecture which is different from the feedforward neural network models in that they have feedback connections. It processes both static and dynamic data like images and videos respectively. This makes LSTM suitable for processing sequential data such as hand sign videos. A standard LSTM unit contains input, output and forget gates and a cell. The cell contains the memory of the values over random time intervals and the input, output and forget gates control the information that enters and exits the cell.

### **1.3.2.4 ConvLSTM**

Similar to the LSTM network, the ConvLSTM performs convolutional input transformations and recurrent transformations. So the 3-dimensions of the input is maintained once the data has passed through the cell. It is therefore suitable to use ConvLSTM models for classifying images in a time-sequence, or videos.

### **1.3.3 CLASS PREDICTION AND AUDIO OUTPUT**

The output from the neural network, which is a prediction of the word class the hand sign belongs to, is then converted to a speech signal, which is then outputted through a speaker. The libraries like PyAudio perform the text-to-speech conversion efficiently and thus enabling the complete translation to work.

### **1.4 ORGANIZATION OF THE CHAPTER**

Chapter 2 tells about the Literature survey. Chapter 3 tells about the image processing technology being used to make image classifications. Chapter 4 informs about the hardware components of the project which include camera, Raspberry Pi, and a speaker. Chapter 5 deals with deep learning architecture and algorithms used for video classification. Chapter 6 lays out the results obtained and its significance. Chapter 7 tells about the conclusion and future work.

## CHAPTER 2

### LITERATURE SURVEY

The work on Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison is carried out by Dongxu Li et al (2019) published on the IEEE Winter Conference on Applications of Computer Vision (WACV) which presented a review about the popular methods and algorithms under use for Sign Language Recognition like Temporal 3D-CNNs and Pose Based TGCNs alongside the comparison between some of those popular models. The research on Large-scale Video Classification with Convolutional Neural Networks done by Andrej Karpathy et al (2014) published on IEEE Conference on Computer Vision and Pattern Recognition, provided a look into the performance of Convolutional Neural Networks on video datasets for activity recognition using the Sports-1M dataset. The research based on Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping carried out by Pascal Schneider et al (2019) published on Lecture Notes in Computer Science, Vol. 11531., Springer, Cham., provided information about the performance of One-Nearest-Neighbour with the help of OpenPose and Dynamic Time Warping in the UTD-MHAD dataset.

The work on Fast and Robust Dynamic Hand Gesture Recognition via Key Frames Extraction and Feature Fusion carried out by Hao Tang et al (2019) published on Neurocomputing, Volume 331, showed the performance of Keyframes extraction and Feature Fusion techniques on the Cambridge dataset. Real-Time Hand Gesture Spotting and Recognition using RGB-D Camera and 3D-Convolutional Neural Network, done by Dinh-Son Tran et al (2019), published Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence, provided a results about the efficacy of 3D-CNNs on the

data obtained from RGB-Depth cameras which gave additional depth information. The work on Dynamic Hand Gesture Recognition using 3D-CNN and LSTM with FSM Context-Aware Model, done by Noorkholis Luthfil et al (2019), Sensors (Basel), provided insight into the usage of a combination of 3-Dimensional Convolutional Neural Networks, Long Short-Term Memory and Finite State Machine for hand gesture recognition.

The work on An Efficient Method for Human Hand Gesture Detection and Recognition using Deep Learning Convolutional Neural Networks, done by P.S. Neethu et al (2020), published in Springer Nature, provided information about the use of CNNs for image classification using segmentation filters. It provides insight into image filtering where the hand region is segmented from the whole image using mask images. The research on Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images, done by Aditya Das et al (2018), published at the International Conference on Smart City and Emerging Technology, provided information about the usage of Inception v3 CNN network on static images. The work on Real-time American Sign Language Recognition with Convolutional Neural Networks, done by Brandon Garcia et al (2016), where the GoogLeNet architecture is used for classification of ASL dataset.

The work on Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges and Opportunities, done by Kaixuan Chen et al (2020), provides an overview of the activity recognition using sensor-based systems rather than video-based systems. The research on Human Activity Recognition Using Deep Learning: A Survey, done Zhang HB et al (2019), published on the Lecture Notes on Data Engineering and Communications Technologies, Volume 52, provides insights into the state of the art for human activity recognition and compares various methods. The work on Human activity recognition using magnetic induction-based motion signals and deep recurrent neural networks,

carried out by Negar Golestani et al (2020), provides information about recognizing human physical activities using wireless sensor networks and magnetic induction systems along with machine learning techniques to detect human motion. The work on Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting done by Xingjian Shi et al (2015), published in the NeurIPS, presented a review on the ConvLSTM network and a comparison of performance with a Fully-Connected LSTM.

## **CHAPTER 3**

### **COMPUTER VISION**

#### **3.1 INTRODUCTION**

Computer Vision is a field of study where machines are trained to interpret and understand images and videos, along with their underlying context. Computer Vision in general, aims to mimic the human visual capabilities using a computer. CV covers the process of acquiring, processing, analyzing and interpreting visual data to produce helpful outputs. It has found its use case in many diverse fields such as solid-state physics, neurobiology, signal processing, robotics, etc. Sub-domains of the computer vision include,

- Object Recognition
- Pose Estimation
- Motion Estimation
- Event Detection
- Learning

##### **3.1.1 APPLICATIONS**

1. The quality and produce amount for crops like wheat and rice are used to identify the food security stability. Computer Vision applications allow continuous and non-destructive monitoring of growth of crops and managing requirements for nutrients.
2. It is used in the medical industry for identifying tumours, organ dimension measurements, flow of blood, etc.

3. CV is used in the military for figuring out enemy vehicles, soldiers, and aids missile guidance.
4. It is essential in autonomous vehicle applications like, submersibles, rovers, Unmanned Aerial Vehicles, etc.

### **3.2 IMAGE PROCESSING**

Image processing technique is the application of algorithms to images so that the images are transformed and useful information is obtained from it . The input and output during processing may be images or image characteristics. Pre-processing is one form of image processing, where the aim is to prepare the images such that they are suitable for an algorithm to be performed on them. Image processing is a subset of computer vision. Image processing is used hand-in-hand with other machine learning algorithms, in order to achieve computer vision.

There are three basic phases in image processing. They are,

- Data collection in the form of images
- Image analysis and manipulation
- Output

Image processing techniques done to two-dimensional visual data like images and videos. Various functions are used for filtering and enhancing the images, so that information could be extracted from them.

Initially image processing was used to improve the quality of the lower quality images. But, due to constraints of the computing equipment during the early days the cost of processing was high. After the development of dedicated



hardware and powerful computers the images could be processed in real-time and even cheaper.

### **3.3 FEATURES**

Features are problem dependent characteristics that can be stated as an interesting or a required part/region of the image. It is the starting point of the various CV pipelines. Features play an important role in every algorithm and hence the final algorithm's efficacy is reliant on the feature detectors' efficiency. Thus desirable property for a feature detector is repeatability, that is, if feature detectors can identify the same features in various images.

#### **Types of Features**

##### **1. Edges**

The edge in general is an arbitrary shape of an object. In practice, it is a set of points in the images with a higher magnitude in gradients.

##### **2. Corners**

It is a point-like feature in the frame of an image that is a 2D matrix.

##### **3. Blobs**

Blob provides a supporting representation of the image structure based on the region. It detects the area in the image which is very smooth that cannot be detected by the corners.

##### **4. Ridges**

Ridges are 1D curves that denote symmetry axes.

### **3.4 FEATURE DETECTION**

Feature detection is a low-level operation or the first operation performed on the matrix of images and it examines every pixel to identify the presence of the features in the pixels. The input is smoothed by a Gaussian kernel, as a mandatory requirement for feature detection, and feature images are calculated and expressed as local image derivative functions. When feature detection requires high computer resources and time, a high-level algorithm is used to continue the flow of data to the feature detection phase, so that the algorithm searches for features in specific regions, masking out unwanted regions. Various CV methods use feature detection in their first step and leads to development of numerous feature detectors. They differ based on the features detected, repeatability the computational complexity.

### **3.5 FEATURE EXTRACTION**

Feature extraction represents the shape information contained in data to ease the classification. In feature extraction the resources required to represent data are reduced. Thus, the main goal of feature extraction is to represent that information in a low dimension space and obtain the necessary information from the data. When the input contains huge redundancies that lead to a large data, it is reduced to a set of features called feature vectors. The transformation of the input data into the set of features is called feature extraction.

If the features extracted are selected diligently, the amount of processing done is significantly reduced, because the entire image does not need to be processed. Pattern recognition is one of the emerging fields of research in the area of image processing. The application of image processing is character recognition, health insurance, document verification, loan, extracting information from cheques,

applications for credit cards, reading bank deposit slips, data entry, tax forms, check sorting, etc.

Feature extraction is an essential phase in pattern classification and aims at the extraction of the necessary data that represents each class. In this process, the required features are obtained from data to produce feature vectors. These feature vectors help the classifiers to identify the input with the target output. Thus, choosing a proper feature extraction technique corresponding to the input requires care.

In our proposed solution, a Deep Neural Network is used for the extraction of features from the images. The network is trained such that the neural network figures out the best features to extract in order to identify and classify the inputs it is provided with. The training occurs based on the dataset provided.

### **3.6 DATASET**

Dataset is a collection of data with an input and output relation. This is considered as a single unit by the machine. Datasets are used for extracting the expected features from the image. There are different types of datasets that can be used for learning. It could contain labeled images in their specific class folders or it could be a document containing the path to the image alongside its class. Dataset plays an important role in the setting up of the expected feature. The efficacy in application of the approaches can be evaluated on the dataset, the larger the dataset the better is the model to be built.

Name	Date modified	Type	Size
test	3/17/2021 11:40 AM	File folder	
train	3/17/2021 11:40 AM	File folder	

Name	Date modified	Type	Size
friend	3/17/2021 11:40 AM	File folder	
hello	3/17/2021 12:35 PM	File folder	
no	3/17/2021 11:40 AM	File folder	
noaction	3/22/2021 3:39 PM	File folder	
yes	3/17/2021 11:40 AM	File folder	



**Fig. 3.1 Representation of labelled data**

A set of videos were recorded for creation of the dataset, which represented four classes representing words frequently used, expressed in the American Sign Language and one class for identifying the absence of any meaningful gesture

being performed. The classes thus comprise the words "Hello", "Yes", "No", "Friend" and "No Action".

The videos are split into train and test sets for performing validation of the dataset. There are about 50 videos for each class in the train dataset and about 15 videos for each class in the test dataset. A total of about 330 videos are used for the entire training process.

## CHAPTER 4

### HARDWARE DESCRIPTION

#### 4.1 RASPBERRY PI

The Raspberry Pi is a single-board computer, that is small-sized, and by the Raspberry Pi Foundation with the main purpose of developing the teaching of basic computing in schools.

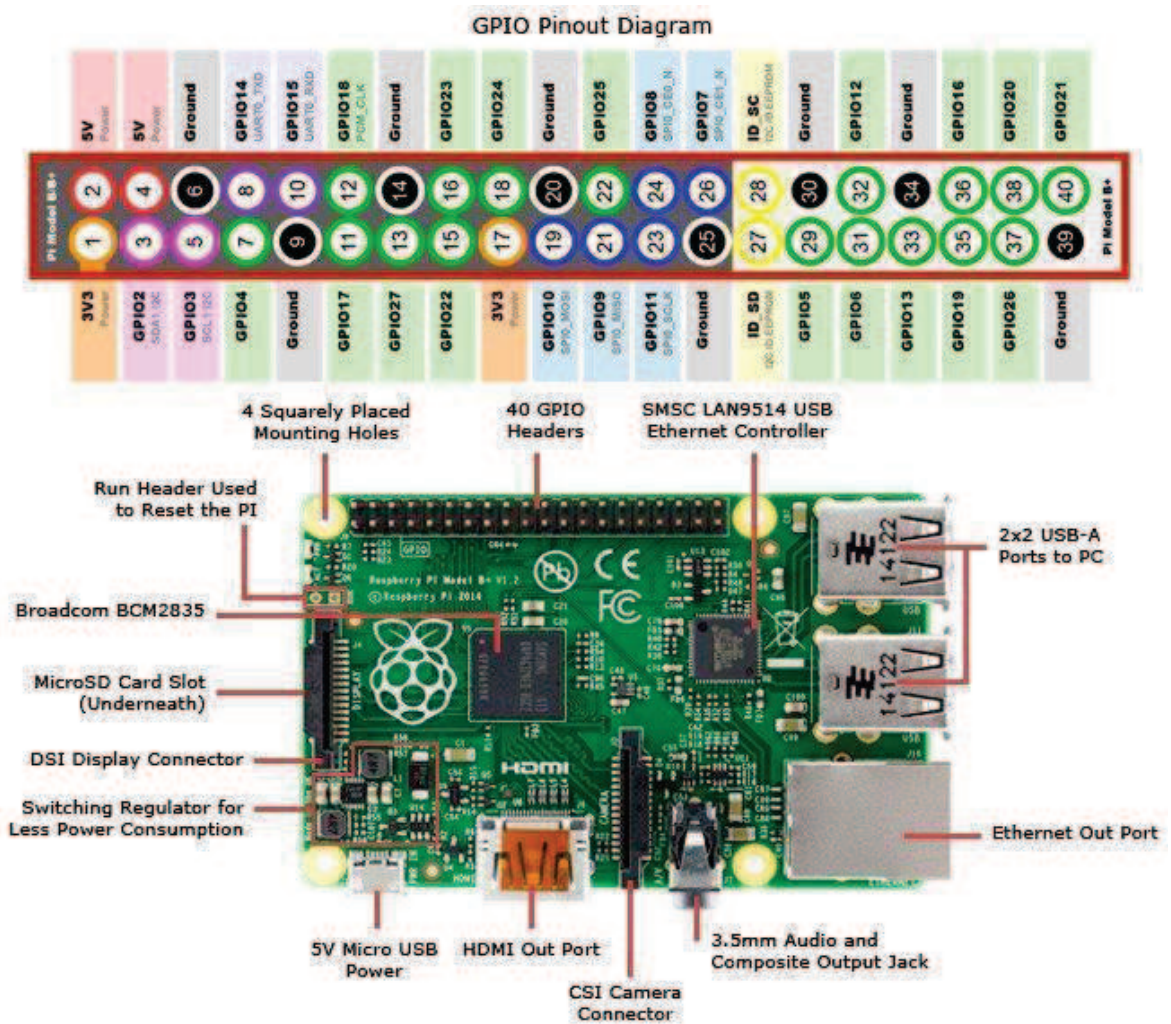
Broadcom BCM2835, a System-on-Chip that contains an ARM1176JZFS with floating point, running at 700 MHz, and a Video core 4 GPU makes this computer so small and powerful. The GPU affords hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode, OpenGL ES 2.0 and is capable of 1Gpixel/s, 1.5Gtexel/s, or 24 GFLOPs of a general-purpose computer. It comes with Model A, Model B, and Model B+.

**Table 4.1 Comparison of Model A, Model B, and Model B+**

<b>Model</b>	<b>RAM</b>	<b>USB Sockets</b>	<b>Ethernet Ports</b>
A	256 MB	1	No
B	512 MB	2	Yes
B+	1 GB	4	Yes

### 4.1.1 Hardware

The operating system must use network time as it does not come with a real-time clock.



**Fig. 4.1 Raspberry pi model B+ and its pin configuration**

The Raspberry Pi is a device that consists of program memory (RAM), processor and graphics chip, Xbee socket, UART, Ethernet port, GPIO pins, power source connector, CPU, GPU, and various interfaces for other external devices. It also requires an SD flash memory card to boot.

## **Power source**

This device consumes 700mA or 3W or power. It can be powered by a Micro USB charger or the GPIO header. The work of powering the Pi can be done by any good smartphone charger.

## **SD Card**

There is in-built storage in the Pi. The operating system is stored on an SD card which can be used with the Pi's SD card slot. The operating system is stored through a card reader.

## **CPU (Central Processing Unit)**

The Central processing unit is the brain of the raspberry pi board which is responsible for carrying out the instructions of the computer through logical and arithmetical operations. The Raspberry Pi 3 Model B uses a Broadcom BCM2837 SoC with a 1.4 GHz 64-bit quad-core [ARM Cortex-A53](#) processor, with 512 KB shared L2 cache. ARM11 series processor is used by the raspberry pi. The Raspberry Pi chip is operating at 700 MHz and it will not become hot enough to need a heatsink or special cooling.

## **GPU (Graphics Processing Unit)**

The GPU is a specially designed chip in the raspberry pi board and that is designed to speed up the operation of image calculations. It is also designed with a Broadcom video-core IV and it supports OpenGL.

## **USB 2.0 Port**



USB 2.0 ports are the means to connect input devices such as a mouse, keyboard to the Raspberry pi. There is only one port on model A, two on model B, and four on model B+. The number of ports can be increased by using an external power USB hub switch which is available as a standard Pi accessory.

### **Ethernet port**

The main way for communicating with additional devices is the ethernet port. This port is used to connect the Pi to the internet.

### **GPIO Pins**

General-purpose input/output (GPIO) pins are standard I/O pins in ICs whose characteristics, including whether input or output mode, can be controlled at run time by the user.

The GPIO pins are used to associate the controller with the other electronic boards. The pins are coded as input and output in the program. The Pi has digital GPIO pins. The pins connect to various electronic components like ultrasonic sensors and cameras to transmit data in digital format as well. GPIO pins can be set as input or output, can be enabled/disabled, inputs are readable (typically high=1, low=0) and outputs are writable/readable.

#### **4.1.1.1 Model B+ Specifications**

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC at 1.40 GHz
- 1.0 GB LPDDR2 SDRAM
- 2.40 GHz and 5.0GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE

- Gigabit Ethernet over USB 2.0 (maximum throughput of 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port
- DSI display port
- 4-pole stereo output and composite video port
- Micro SD port
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support

#### **4.1.2 Need for Raspberry pi**

- The main advantage of Raspberry Pi is its small size and relatively high power.
- One other advantage of Raspberry Pi is that it speeds up solving problems using hardware compared to the software implementation.
- It also makes use of parallelization.
- Lower power consumption.

#### **4.1.3 Setting up Raspberry pi**

The Raspberry Pi is not generally provided with an operating system. To set up Raspberry Pi,

- Download an ISO disk image from the raspberrypi.org site and then extract the folder contained in the zip file to some convenient location.

- Use the Raspberry Pi imager to install the Raspbian on the SD card.
- Wait till the Raspberry Pi Imager to finish writing into the SD card.
- Eject the SD card safely after writing.
- Insert the SD card and set it up with Raspbian into the microSD card slot on the bottom of the Raspberry Pi.
- Turn on the USB power supply and connect it to the Raspberry Pi's power port.
- After Raspbian has been installed, click OK. Now Raspberry Pi will restart, and Raspbian will then boot up.
- After some time the Raspbian Desktop will appear.

## 4.2 CAMERA

Logitech Webcam C170 is used here. Logitech's Fluid Crystal Technology is used and the VGA sensor allows video calling at 30 fps with 640 X 480 resolution. It can capture video at 1024 X 768 resolution. It captures images at 5 megapixels. This camera captures the surrounding objects and these images are processed by using Deep learning algorithms. Any custom camera that could reduce the size of the device could be used. This camera is used for the sake of ease of implementation.



**Fig. 4.2 Logitech webcam C170**

### 4.2.1 System requirement

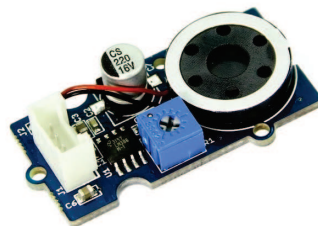
- Processor Speed 1.0 Hz
- Minimum RAM Size 512.0 MB
- Minimum Hard Drive Space 200.00 MB

#### 4.2.2 Features

- Video Captures at 320x240, 640x480, 1024x768
- Optical Resolution: 640x480
- Size is 21.30 x 15.50 x 7.90 cm
- Weight of 0.5lb

#### 4.3 SPEAKER

Grove compatible generic speakers are used for outputting audio signals. It is appropriate for single audio signalling and sound generation. It is equipped with a potentiometer to adjust loudness and to clear sound out of a microcontroller. Different tones can be generated using the speaker with different input frequencies.



**Fig. 4.3 Speaker module**

##### 4.3.1 Specifications

- Operating Voltage (VDC) - 4 to 5.5
- Voltage Gain (dB) - 46
- Bandwidth (KHz) - 20

- Length (mm) - 130
- Width (mm) - 90
- Height (mm) - 6
- Mounting hole diameter(mm) - 2
- Weight (gm) - 12
- Shipment weight - 0.016 Kg
- Shipment dimensions -  $9.5 \times 6.8 \times 1.2$  cm

## **CHAPTER 5**

## **DEEP LEARNING**

### **5.1 EVOLUTION OF DEEP LEARNING**

Machine Learning is a subset of artificial intelligence associated with creation of algorithms that can modify themselves without human intervention to get the desired result with the use of structured data. These ML algorithms are designed to learn, by understanding labeled data and then use it to produce new results with more datasets. Machine Learning algorithms build a mathematical model based on sample data in order to make predictions or decisions without being explicitly programmed to perform the task. There are many machine learning approaches such as supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning.

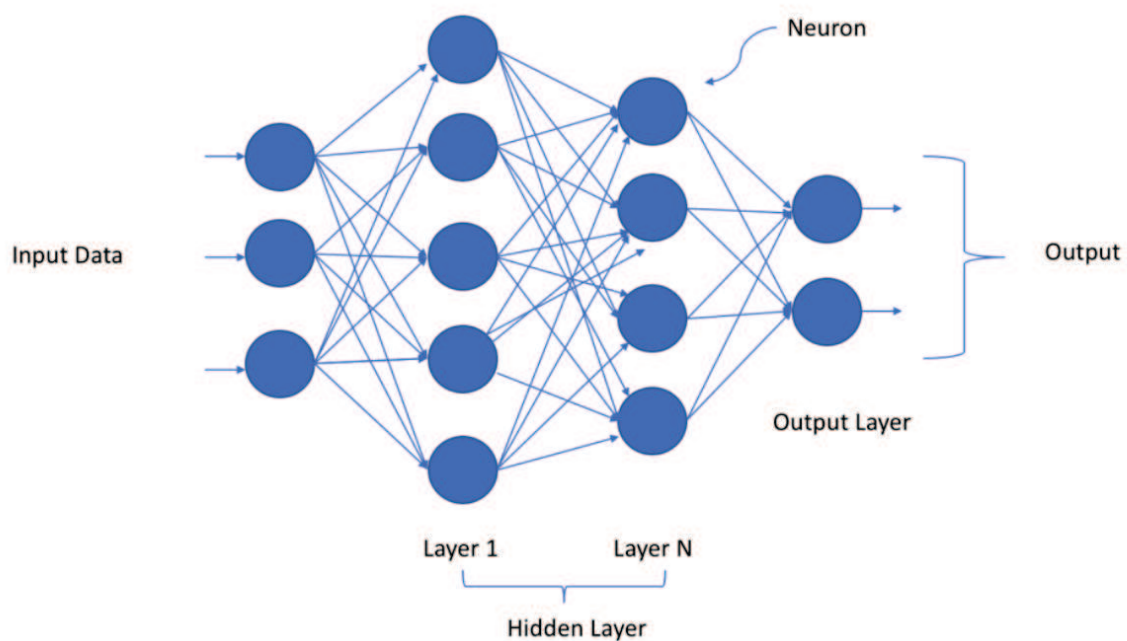
However, they need to be retrained through human intervention when the actual output isn't the desired one. Since there were many limitations in Machine learning algorithms, one of the algorithms which brought a revolution in modern society is Deep Learning. Deep Learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstractions. These methods have dramatically improved the state of the art in speech recognition, visual object detection, object detection, and many other domains too.

Deep Learning is a subset of machine learning where algorithms are created and function similarly to machine learning, but there are many levels of these algorithms, each providing a different interpretation of the data it conveys. This network of algorithms is called Artificial Neural Networks. Deep learning networks do not require human intervention, as multilevel layers in neural networks place data in a hierarchy of different concepts, which ultimately learn

from their own mistakes. A deep learning architecture is a multi-layer stack of simple modules, all of which are subject to learning, and many of which compute non-linear input-output mappings.

### 5.1.1 Neural Network

As explained above, Deep learning is a sub-field of Machine learning dealing with algorithms inspired by the structure and function of the brain called ANN. It is similar to our nervous system, where each neuron is connected to other neurons and passing information.



**Fig. 5.1 A basic Neural Network structure**

## 5.2 CONCEPTS OF DEEP LEARNING NEURAL NETWORK

### Linear Regression

It is a statistical method that allows us to summarize and study the relationships between two continuous variables. The output is plotted in a graph as dot points. Based on the points near the straight line, the final output is predicted.

### **Logistic Regression**

It is a statistical method for analyzing a dataset in which one or more independent variables determine an outcome. The outcome is measured in which there are only two possible outcomes: True or False. In logistic regression, similar to linear regression, it will find the best possible straight line that separates the two classifications.

### **Activation Function**

Activation functions are the functions that decide the actual output and therefore output of these layers as its 'activations'. There are many types of activation functions present.

### **Weights**

When input data comes into a neuron, it gets multiplied by a weight value that is assigned to this particular input. These weights start out as random values, and as the neural network learns about the nature of input data it leads to, the network adjusts the weights based on any errors in categorization that the previous weights resulted in and this is mentioned as training. Once the network is trained, it can be used for predicting the output for similar input.

### **Bias**

Both weights and biases are the learnable parameters of the deep learning models. It is a value-added to each layer's output.



## **Forward Propagation**

By propagating from the first layer (input layer) through all the mathematical functions represented by each node, the network outputs a value. This process is called Forward pass.

## **Gradient Descent**

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm. Gradient descent is used to find the minimum error by minimizing a “cost” function.

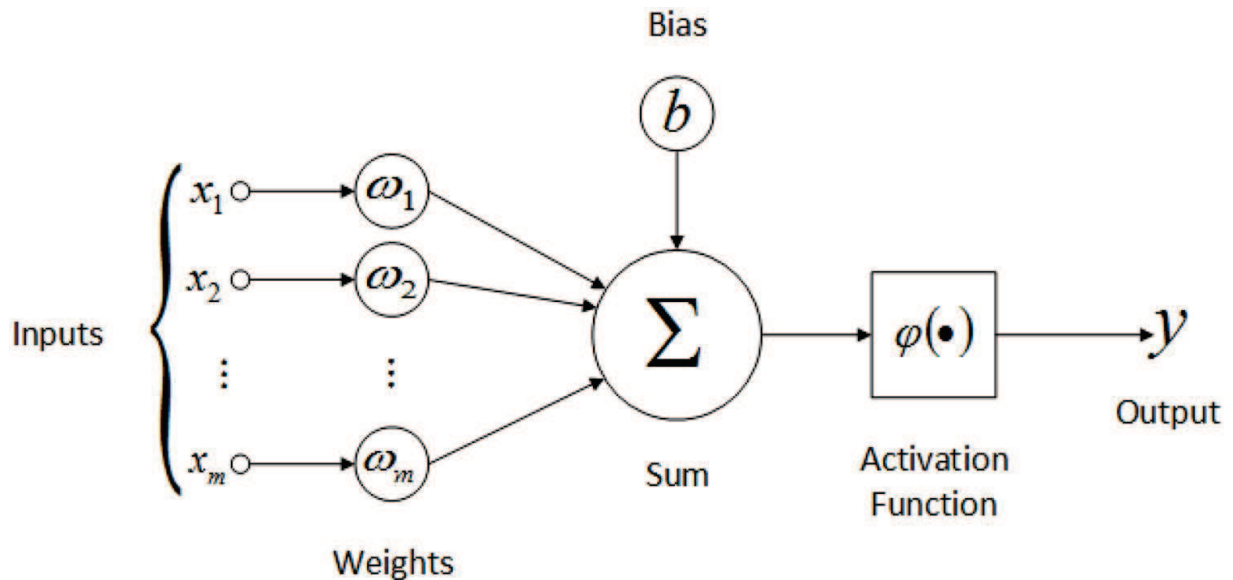
## **Back propagation**

In neural networks, Forward Propagation is done to get the output and it is compared with the real value to get the error. In order to minimize the error, backward propagation is done by finding the derivative of the error with respect to each weight and then subtracting this value from the weight value. This is called Back propagation.

## **Modeling of Artificial Neurons**

It depicts a neuron connected with n other neurons and thus receives n inputs ( $x_1, x_2, \dots, x_n$ ). This configuration is called a Perceptron. The inputs ( $x_1, x_2, \dots$ ) and

the weights ( $w_1, w_2, \dots$ ) are real numbers and can be either positive or negative. The perceptron consists of weights, a summation processor, and an activation function.



**Fig. 5.2 Modelling of Artificial Neurons**

One particular type of deep, feedforward network that was much easier to train and generalized much better than networks with full connectivity between adjacent layers. This was the Convolutional Neural Network (ConvNet).

### 5.3 CONVOLUTIONAL NEURAL NETWORK (CONVNET)

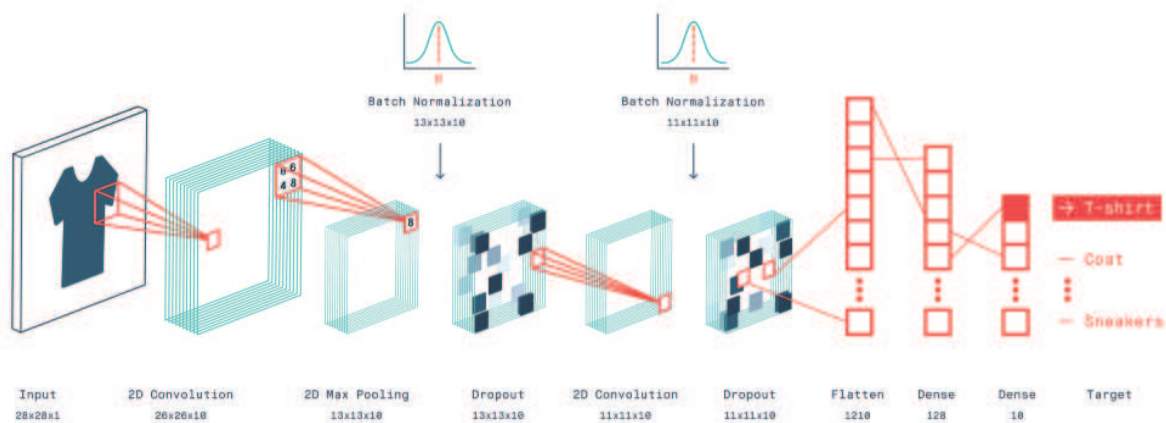
In neural networks, Convolutional Neural Network is one of the main categories to perform image recognition, image classification, object detection,

recognizing faces, etc,.. are some of the areas where CNN's are widely used. ConvNets are designed to process data that come in the form of multiple arrays, for example, a color image composed of three 2D arrays containing pixel intensities in the three color channels. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, 2D for images, and 3D for video or volumetric images.

The architecture of a typical ConvNet is structured as a series of stages. The first few stages are composed of convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter. The result of this locally weighted sum is then passed through a non-linearity function. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. Mathematically, the filtering operation performed by a feature map is a discrete convolution. Also, if the pixel size is large, learning and computation will be more. Therefore, the convolution operation is made so that large pixels can be detected and hence the name convolution.

Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically several features into one. A typical pooling unit computes the maximum of a local patch of units in one feature map. Neighboring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation. Two or three stages of convolution, non-linearity, and pooling are stacked, followed by more convolutional and fully-connected layers. Backpropagating gradients through a ConvNet is simple as

through a regular deep network, allowing all the weights in all the filter banks to be trained.



**Fig. 5.3 An example of a CNN architecture**

## 5.4 LAYERS OF CNN

CNN's image classification takes an input image, processes it, and classifies it under various categories depending on the dataset it is being trained on. The computer sees an image as an array of pixels and it depends on the image resolution. Technically, Deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (kernels), pooling and fully connected layers, and a classification layer.

### 5.4.1 Convolutional Layer

The convolutional layer is the first layer to extract features from an input image. It preserves the relationship between pixels by learning image features using small squares of input data. Convolution is the mathematical operation that takes two inputs such as an image matrix and a filter or kernel. The image matrix has a dimension of  $h \times w \times d$  and a filter of dimension as  $f_h \times f_w \times d$ . It outputs a

volume dimension of  $(h - f_h + 1) \times (w - f_w + 1) \times 1$ . The output obtained is mentioned as a ‘feature map’. Always, the number of channels in the filter should match the number of channels in the input image.

### Strides

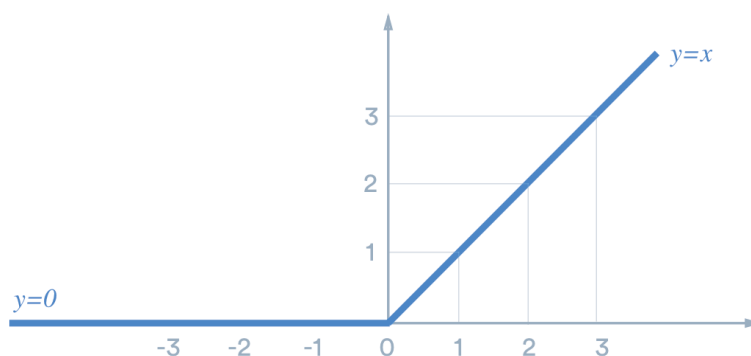
Stride is the number of pixels that shift over the input matrix. For example, if the stride is 1, then the filter moves 1 pixel at a time.

### Padding

Sometimes in the convolution process, output shrinks at each layer and a lot of information is lost. Also, there is a possibility where the filter does not perfectly fit the input image. There are two types: valid convolution and same convolution. In “valid” convolution, no padding occurs and only keeps the valid part of the image. In “same” convolution, padding occurs such that the output size of the image should be the same as that of the input image.

### Non-Linearity function

ReLU stands for Rectified Linear Unit for a non-linear operation. The important usage of the ReLU function is that ConvNet should learn non-negative linear values. The output is  $f(x) = \max(0, x)$ . Though there are many nonlinear functions, ReLU’s performance is better than other functions.



## **Fig. 5.4 Example for ReLU function**

### **5.4.2 Pooling Layer**

ConvNets use a pooling layer to reduce the size of the representation, to increase the speed of computation as well as make some features that detect a bit more robust. This layer would reduce the number of parameters when the image size is too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains some important information. It is of different types:

#### **Max Pooling**

Max Pooling takes the largest element from the rectified feature map. This type of pooling is generally used in all architectures.

#### **Min Pooling**

Min Pooling takes the smallest element from the rectified feature map in accordance with the filter size.

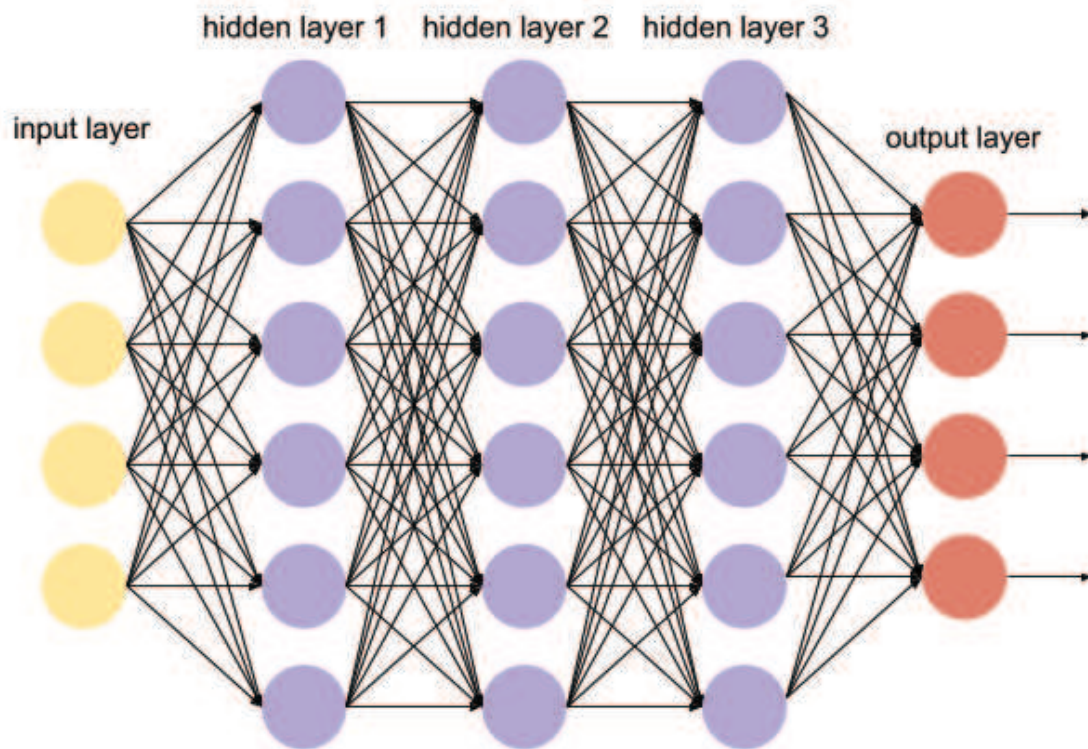
#### **Average Pooling**

Average pooling returns the average of all the values from the portion of the image covered by the kernel.

### **5.4.3 Fully Connected Layer**

After several hidden layers, the high-level reasoning in the neural network is done via Fully-connected layers. The fully Connected Layer involves weights, biases, and neurons. Here, neurons in one layer are connected to neurons in another

layer. In this FC layer, the matrix is flattened into a vector form and fed to a neural network. The below image is an example of a flattened FC layer.



**Fig. 5.5 Example for Fully Connected Layer**

The matrix is flattened into a vector form as  $x_1, x_2, x_3, x_4$  and the output consists of three classes such as  $y_1, y_2$  and  $y_3$ . Also, backpropagation is applied to every iteration of training. Over a series of epochs, the model is able to identify the dominant features and is classified using the SoftMax layer. This SoftMax layer is a multi-class classifier that classifies the objects based on training datasets and provides a confidence value.

## **5.5 CNN ARCHITECTURE**

There are many architectures used for the implementation of Convolutional Neural Networks. Almost all the CNN architectures follow the same general design principles for processing the input. It can be classified into Classic network architectures and Modern architectures. Classic network architectures include LeNet-5, AlexNet and VGG 16. Modern architectures include GoogleNet, ResNet, etc.

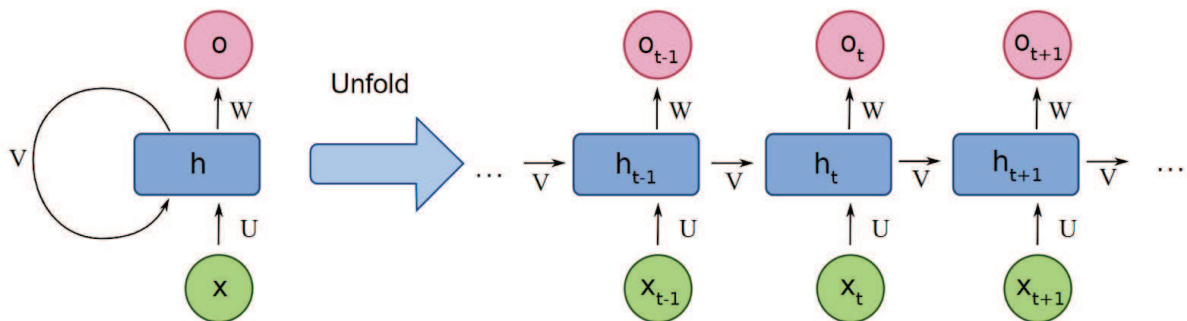
Static images in sign languages are used to represent alphabets and numbers. They are used when the user of sign language has to spell out some special words such as names or express numbers. But when it comes to representing words, the expressions are not static, rather they require movement of the hand, which is captured in the form of videos. Videos are represented as a sequence of frames or images. They are essentially 3-dimensional arrays composed of 2D images spread over time.

Although CNNs perform well with images, their ability is limited to extraction of spatial features within the image, i.e., only features in 2 dimensions. When working with videos, extracting temporal features, i.e., time-related features along with the spatial features is necessary. This is because videos are a collection of images or frames over time, i.e., they are a time-distributed sequence of frames. Along with the features present within an image, the inter-relations between the different images in the sequence should be extracted. 2D-CNNs, while being able to capture the spatial features within the image in a video, does not extract the temporal information that is provided by the sequence. Hence, 2D-CNNs are not the most suitable if we have to extract the time-spaced information between the frames of a video sequence.

## **5.6 RECURRENT NEURAL NETWORK (RNN)**

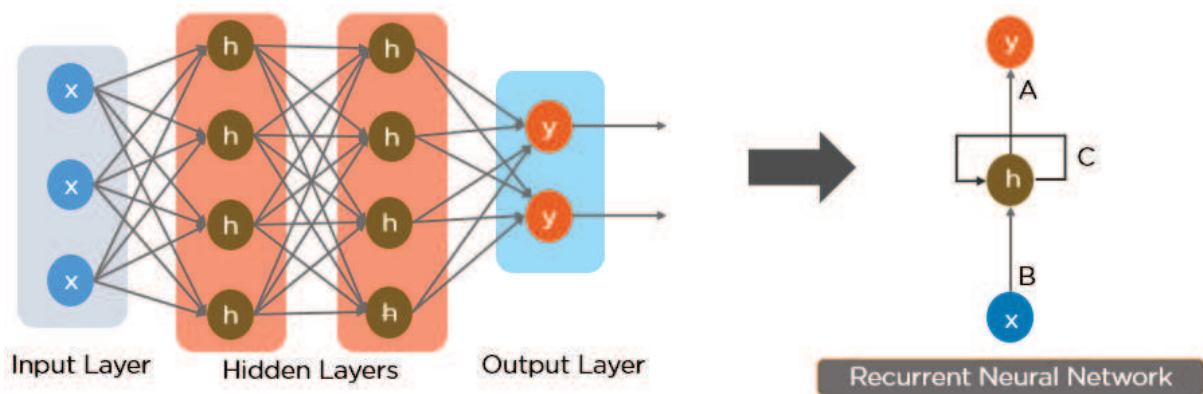


A recurrent neural network is another neural network architecture where the connections between nodes is in the form of a directed graph along a temporal sequence. RNNs can use their internal state to process variable length sequences of inputs. This internal state or memory enables them to take information from prior inputs to affect the current input and output. Recurrent Neural Networks use backpropagation through time (BPTT) algorithm to determine the gradients, which is different from traditional backpropagation as it is specific to sequence data.



**Fig. 5.6 Recurrent Neural Network architecture**

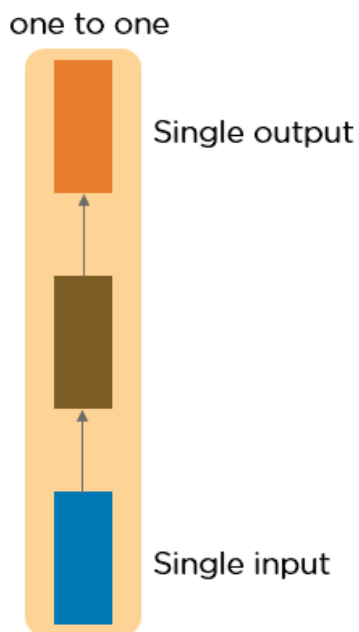
A feed-forward neural network can be converted into a RNN as shown below.



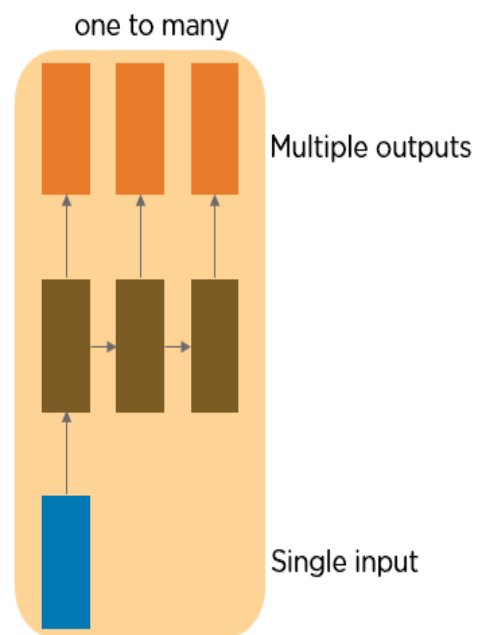
**Fig. 5.7 Composition of an RNN from a dense neural network**

The nodes in the different layers of the neural network are combined to form a single layer of RNN, with network parameters A, B and C.

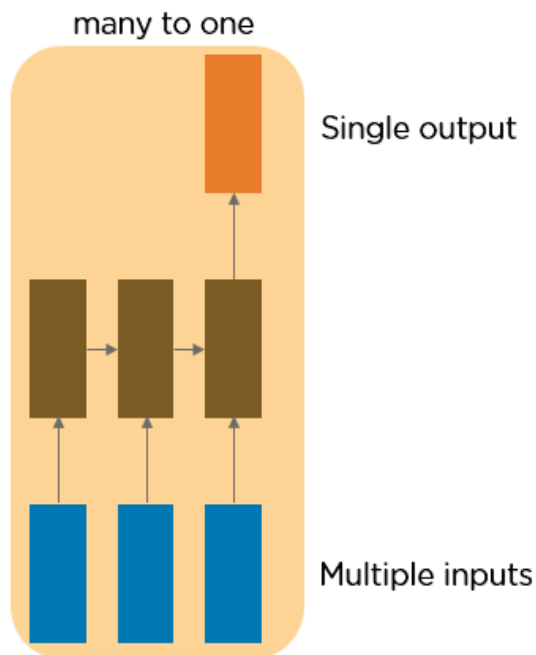
There are four types of RNN, which are shown below.



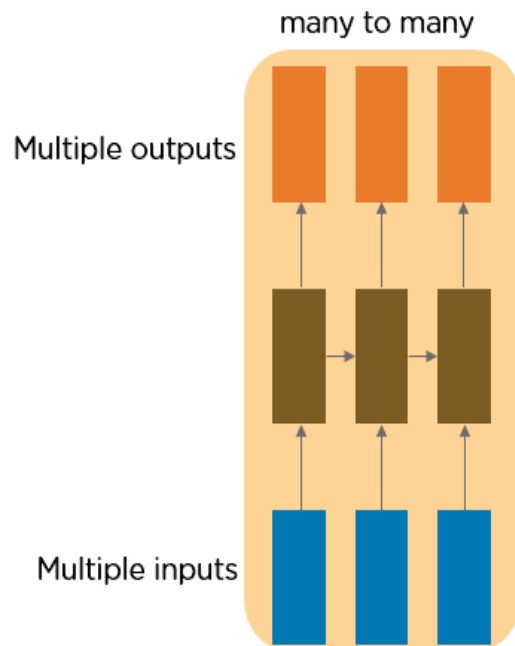
**Fig. 5.8 (a) One to One**



**Fig. 5.8 (b) One to Many**



**Fig. 5.8 (c) Many to One**

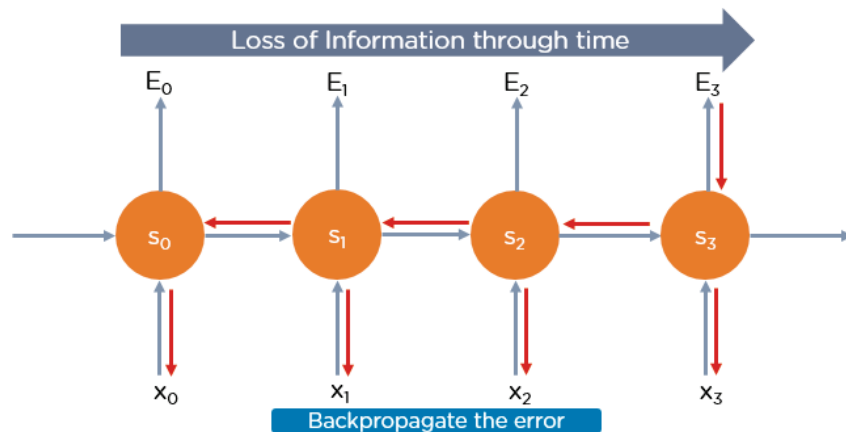


**Fig. 5.8 (d) Many to Many**

RNNs have two common issues: vanishing gradient and exploding gradient.

### 5.6.1 Vanishing Gradient Problem

The gradients carry information used in the RNN, and when the gradient becomes too small, the parameter updates become insignificant. No useful gradient information from the output end of the model is propagated to the layers near the input of the model. Longer the sequence is, the smaller the gradient tends to become. This makes the learning of long data sequences difficult. The models become unable to learn on a given dataset, or prematurely converge to a non-optimal solution.



**Fig. 5.9 Represents the possibility of losing information due to vanishing gradient**

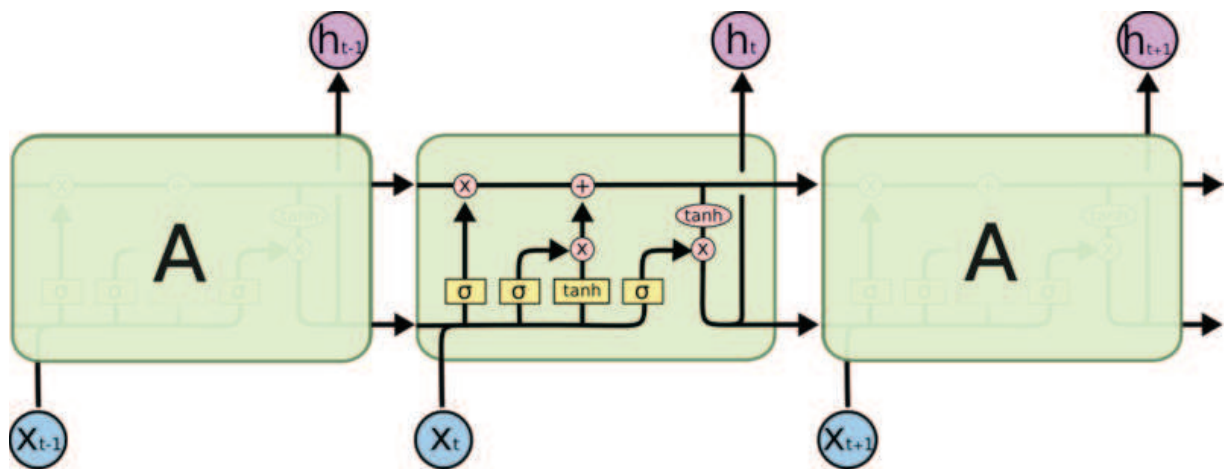
### 5.6.2 Exploding Gradient Problem

While training a neural network, if the slope tends to grow exponentially instead of decaying, this is called an Exploding Gradient. This problem arises when large error gradients accumulate, resulting in very large updates to the neural network model weights during the training process.

Long training time, poor performance, and bad accuracy are the major issues in gradient problems.

### 5.7 LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory is an RNN architecture that has feedback connections, unlike the standard feedforward neural networks. The model passes the previous hidden state to the next step of the sequence, which allows the network to include the previous data in the decision making process.



**Fig 5.10 Long Short-Term Memory architecture**

A common LSTM unit contains a cell, an input gate, an output gate and a forget gate. The cell retains values over time intervals and the gates regulate the flow of information in and out of the cell. They solve the vanishing gradient problem faced by vanilla RNNs, because LSTM units allow gradients to flow unchanged.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

The LSTMs have the ability to remove or add information to the cell state, regulated by structures called gates. Gates allow to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

### **5.7.1.1 Forget Gate**

The forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

### **5.7.1.2 Input Gate**

To update the cell state, we have the input gate. First, the previous hidden state and current input are passed into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 corresponds to values being unimportant, and 1 corresponding to values being important. The hidden state and current input are passed into the tanh function to squish values between -1 and 1 to help regulate the network. Then the tanh output is multiplied with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.

### **5.7.1.3 Cell State**

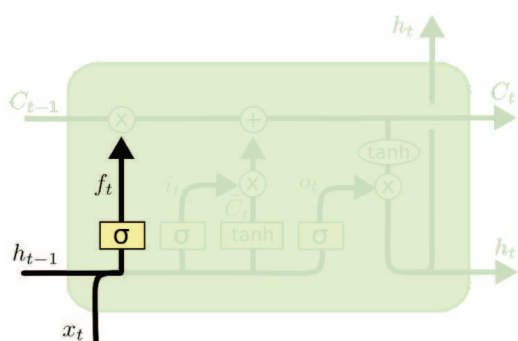
To calculate the cell state, first, the cell state gets pointwise multiplied by the forget vector. Values in the cell state could be dropped if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives a new cell state.

### **5.7.1.4 Output Gate**

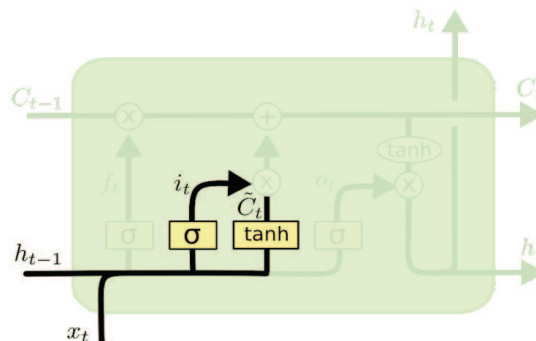
The output gate decides what the next hidden state should be. The hidden state contains information on previous inputs and is also used for predictions. First, the previous hidden state and the current input are passed into a sigmoid function.

Then the newly modified cell state is passed to the tanh function. The tanh output is multiplied with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

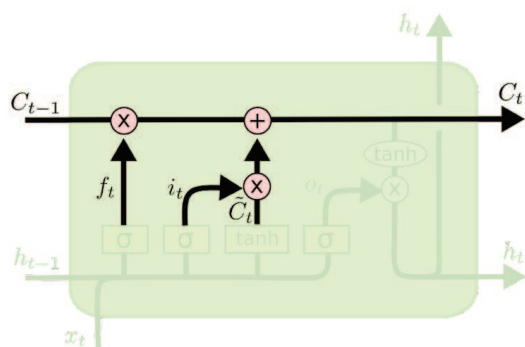
The steps taken under one LSTM cell are shown below.



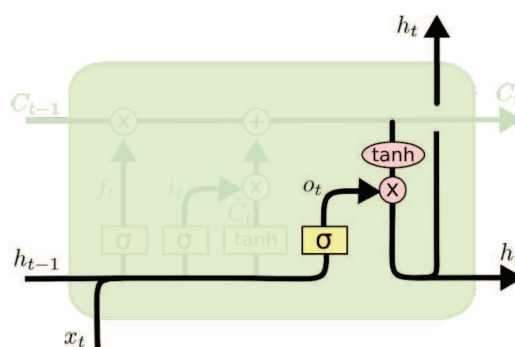
**Fig. 5.11 (a) Forget gate decision**



**Fig. 5.11 (b) Input gate decision**



**Fig. 5.11 (c) Updating the old cell state into the new cell state**



**Fig. 5.11 (d) Output gate decision**

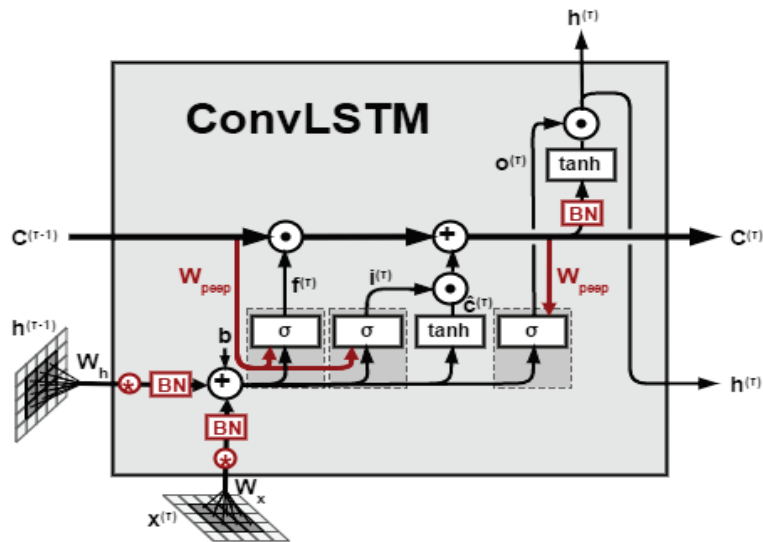
## 5.8 CONVLSTM LAYER

ConvLSTM layers are similar to the LSTM layer, except the internal matrix multiplications within the images are exchanged with convolution operations, which tend to work well for image data. Although a Fully Connected LSTM layer

is powerful for handling temporal correlation, it contains too much redundancy for spatial data. To avoid this ConvLSTMs, which possesses convolutional structures in both input-to-state and state-to-state transitions can be used in place.

Some of the important equations are,

$$\begin{aligned}
 i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
 f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
 C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
 o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
 H_t &= o_t \circ \tanh(C_t) \quad []
 \end{aligned}$$



**Fig. 5.12 Structure of a ConvLSTM network.**

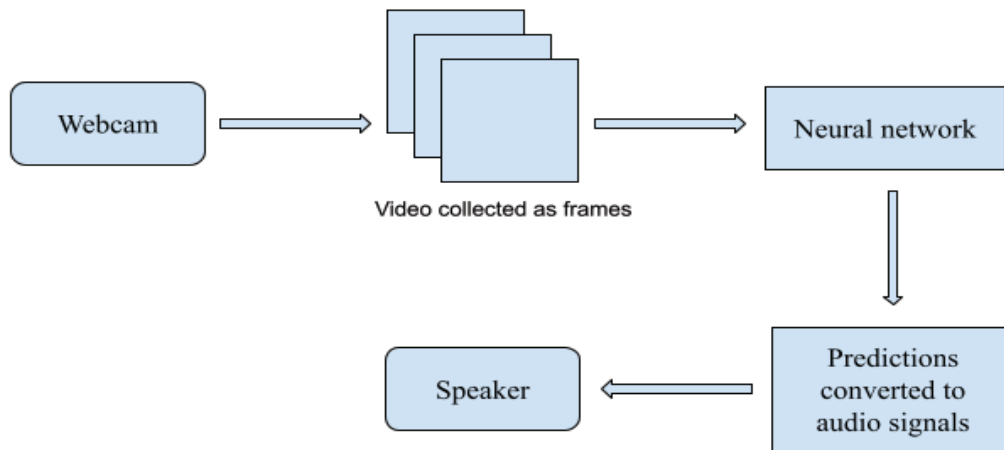
Due to their effectiveness on image-based time sequences a ConvLSTM neural network was chosen for this project.



## CHAPTER 6

### PROPOSED METHODOLOGY

Sign Language translation is split into three stages, namely, video capturing, prediction, and conversion of text-to-speech so that audio signals can be outputted. The pipeline involves the usage of camera and speakers, interfaced with the microcontroller, Raspberry Pi.



**Fig. 6.1 Block diagram of the recognition and translation process**

#### STEPS:

1. The camera interfaced with the Raspberry Pi captures the video frame-by-frame.
2. The frames are stored in a buffer for a certain duration.
3. The frames in the buffer are then passed onto the neural network for prediction of classes.
4. Outputting the predicted class in the form of audio signals through the speaker using text-to-speech conversion.

## **6.1 CAMERA**

The camera connected to the microcontroller through USB, captures the video frame-by-frame, which is stored in a buffer that can hold 80 frames. The frames are captured at 30 frames per second and each image is resized to 64 x 64 pixels. Therefore the input is in the shape of (80, 64, 64) representing the number of frames, image height and width respectively.

Resizing of images is done to reduce the load on the network while training and processing, as much of the spatial features remain the same even after the resolution is reduced multifold. 80 frame buffer is chosen to normalize the video sequence length, which is usually varies due to the 2 - 4 seconds of clip time corresponding to the gesture being shown, to a constant value.

OpenCV library is used for capturing the image in frames. OpenCV is an open source, computer vision library that contains various functions for performing image processing and video processing operations. This makes it suitable for gesture recognition.

## **6.2 PROPOSED DEEP LEARNING ALGORITHM**

A ConvLSTM deep neural network is used for predicting the classes from the given input video sequence. Tensorflow library is used for performing the neural network related operations. Other data handling libraries such as numpy and sci-kit learn are used to aid in the training process. The input should be made compatible with the neural network and hence the appropriate buffer is used to store the frames. A multi-class classification is performed on the data, so as to choose 1 from the 5 given classes. Thus, after prediction the class the video is corresponding to can be obtained.

Layer (type)	Output Shape	Param #
conv_lst_m2d_4 (ConvLSTM2D)	(None, 80, 64, 64, 64)	154624
batch_normalization_4 (Batch Normalization)	(None, 80, 64, 64, 64)	256
conv_lst_m2d_5 (ConvLSTM2D)	(None, 64, 64, 8)	20768
batch_normalization_5 (Batch Normalization)	(None, 64, 64, 8)	32
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 128)	4194432
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 5)	645

**Fig. 6.2 The proposed neural network architecture**

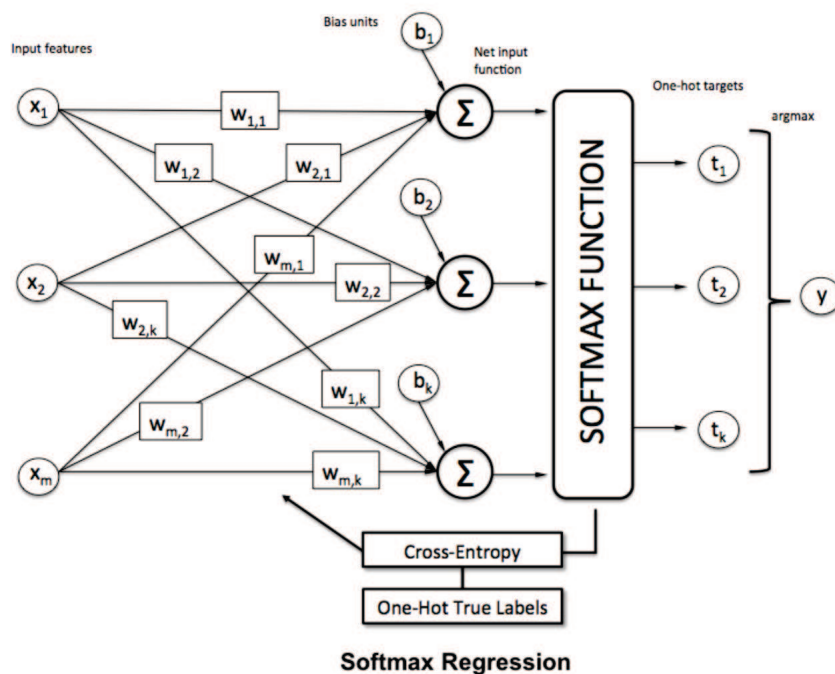
### 6.2.1 ConvLSTM Layer

A ConvLSTM layer is similar to a regular LSTM layer except that the internal matrix multiplications are exchanged for convolution operations. The ConvLSTM layer accepts a 4-dimensional input of (number of frames, image height, image width, number of filters). There are two ConvLSTM layers used with the first one having 64 convolution filters and the second one having 8 filters. This results in the first layer having an input shape of (80, 64, 64, 64) and the second layer having an input shape of (80, 64, 64, 8). Since “same” padding is used the dimensions of the image are retained for the first two layers involving convolutions. The return\_sequences is made true for the first ConvLSTM layer so that it can pass the 80 frames given to it as input, to the next ConvLSTM layer.

Batch normalization is applied after each ConvLSTM layer in order to standardize the input to a layer for every mini-batch, thus stabilizing the learning process and reducing the number of epochs required to train the network.

### 6.2.2 Dense layer

Once the ConvLSTM layers extract the spatial and temporal features within the input, the neural network implements a flatten layer to create a 1-dimensional array for further processing with the dense layers. The dense layer consists of neurons that are interconnected to all the input and output neurons. This dense layer allows the extracted features to be used in the prediction of classes. The hidden layer contains 128 neurons and they are used with dropout to avoid the problem of overfitting. Finally, these neurons are connected to 5 output neurons representing the classes contained in the data. Since multi-class classification is used, the output layer uses a SoftMax function as activation and the class with the highest probability is chosen as the correct class.



**Fig. 6.3** An example of the softmax function

### **6.3 AUDIO OUTPUT**

Once the neural network predicts the class of the video sequence the class text is converted into sound signal. This is attained with the help of the Python-Text-To-Speech library. The audio is outputted through the speaker connected as a peripheral device to the microcontroller.

The engine object of the pyttsx3 library is used to set the rate, volume and voice of the instance running. This helps to customize our output corresponding to the situation. The analog output is attained using the 3.5mm audio jack.

Once the audio is outputted, the buffer is released and new frames are collected and stored in the buffer. This process continues on, thus enabling translation of the signs performed.

## CHAPTER 7

### RESULT ANALYSIS






Sign language translation, with the advancement in hardware and algorithms to be run on those powerful hardware, could eliminate the gap between the differently-abled and others. A portable device that is able to translate sign language would help the people with hearing and speaking difficulties, in various social and work environments.

This chapter deals with the results obtained at the various stages of the implementation. The dataset is very crucial for improving the ability of a network in learning the required features and ultimately, the performance of the network.

#### 7.1 RESULTS FROM THE COLLECTION OF DATASET

 test	Date modified: 3/22/2021 3:37 PM
 train	Date modified: 3/22/2021 3:32 PM

**Fig. 7.1 Organization of the dataset into train and test sets**

 friend	Date modified: 3/17/2021 11:40 AM
 hello	Date modified: 3/17/2021 12:35 PM
 no	Date modified: 3/17/2021 11:40 AM
 noaction	Date modified: 3/22/2021 3:36 PM
 yes	Date modified: 3/17/2021 11:40 AM

**Fig. 7.2 Organization of the classes to be trained**

 0.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 563 KB
 1.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 717 KB
 2.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 672 KB
 3.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 663 KB
 4.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 838 KB
 5.mp4 Length: 00:00:01	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 604 KB
 6.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:32 PM Size: 680 KB
 7.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 680 KB
 8.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 840 KB
 9.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 654 KB
 10.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 731 KB
 11.mp4 Length: 00:00:03	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 960 KB
 12.mp4 Length: 00:00:02	Frame height: 480 Frame width: 640	Date modified: 3/16/2021 10:33 PM Size: 811 KB

**Fig. 7.3 Collection of videos of a class**



**Fig. 7.4 A 640 x 480 frame**

**Table 7.1. Organization of the classes representing words**

<b>CLASS</b>	<b>TRAIN DATA</b>	<b>TEST DATA</b>
Hello	50	16
Yes	50	16
No	51	16
Friend	50	15
No Action	50	16

The dataset was captured using the webcam with the help of OpenCV library and stored such that each directory represents a class with the videos of that class inside the directory. Three different people perform signing for the classes. Each video is 2-4 seconds long, with a frame rate of 30 frames per second, and in the ".MP4" format, containing all the three colour channels. Each frame is of the size 64 x 64 pixels. The train dataset consists of about 50 videos representing each class, with the test dataset containing about 15 videos for each class. The dataset totally has 251 videos for train data and 79 videos for test data.

## **7.2 RESULTS OBTAINED FROM TRAINING OF THE NEURAL NETWORK**

The neural network is fed with an array of frames as input to the first layer. The shape of the array is thus (80, 64, 64, 64) which corresponds to (number of frames, image height, image width, number of filters for convolution). The output from the neural network is an array with the probabilities of each class, which is calculated by the softmax activation in the output layer. The trained model is exported and stored.



Layer (type)	Output Shape	Param #
conv_lst_m2d_4 (ConvLSTM2D)	(None, 80, 64, 64, 64)	154624
batch_normalization_4 (Batch Normalization)	(None, 80, 64, 64, 64)	256
conv_lst_m2d_5 (ConvLSTM2D)	(None, 64, 64, 8)	20768
batch_normalization_5 (Batch Normalization)	(None, 64, 64, 8)	32
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 128)	4194432
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 5)	645

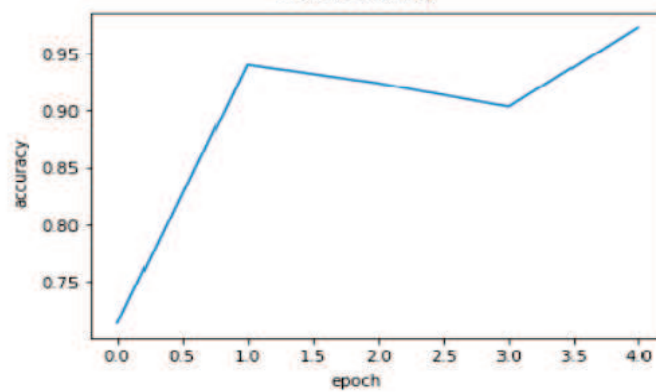
=====  
 Total params: 4,370,757  
 Trainable params: 4,370,613  
 Non-trainable params: 144

**Fig. 7.5 Neural network model along with the number of parameters**

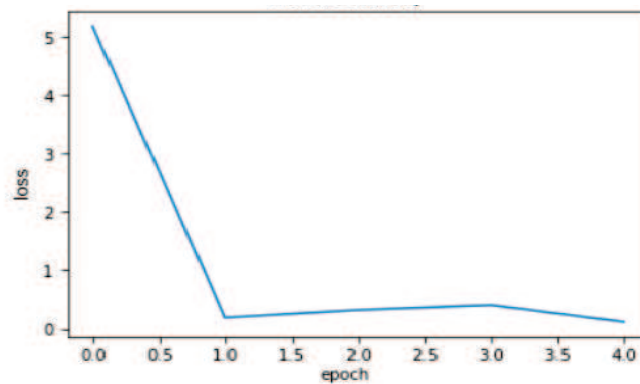
```

Epoch 1/5
31/31 [=====] - 4188s 135s/step - loss: 5.6382 - accuracy: 0.3772 - val_loss: 1.4611 - val_accuracy: 0.4051
Epoch 2/5
31/31 [=====] - 4024s 130s/step - loss: 0.4594 - accuracy: 0.8237 - val_loss: 0.8257 - val_accuracy: 0.5696
Epoch 3/5
31/31 [=====] - 4174s 135s/step - loss: 0.2051 - accuracy: 0.9638 - val_loss: 2.3729 - val_accuracy: 0.4051
Epoch 4/5
31/31 [=====] - 4196s 136s/step - loss: 0.2297 - accuracy: 0.9512 - val_loss: 3.3454 - val_accuracy: 0.5190
Epoch 5/5
31/31 [=====] - 4067s 131s/step - loss: 0.0424 - accuracy: 0.9804 - val_loss: 12.7216 - val_accuracy: 0.4937
INFO:tensorflow:Assets written to: /content/drive/MyDrive/video_model1.model/assets
  
```

**Fig. 7.6 Results after each epoch of the training process**



**Fig. 7.7 (a) Accuracy vs Number of epochs**



**Fig. 7.7 (b) Loss vs Number of epochs**

The accuracy of the training was about 98% and the loss was about 2%.

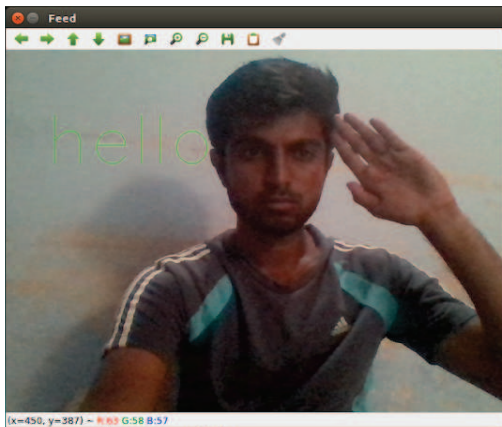
Previous works on American Sign Language use 3-Dimensional CNNs based algorithms or a combination of CNN and RNN architectures in their algorithms. The accuracy obtained on different datasets by using CNN-RNN based algorithm which uses VGG-GRU architecture was about 64% and 3D-CNN based algorithm which uses I3D architecture, was about 90% [2]. The results from experimenting with our dataset produced accuracy of about 98%, with the ConvLSTM architecture in our implementation.

Although this dataset is smaller, it shows that the ConvLSTM networks are a viable approach to this sign language recognition as they learn about the spatial

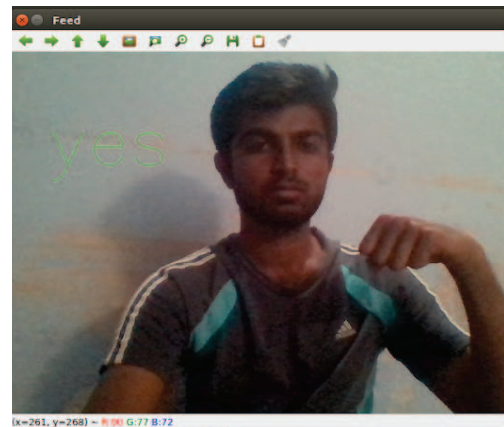
and temporal features simultaneously, in contrast to CNN-RNN networks that extract and learn from the features separately. 3D-CNNs while having more accuracy than CNN-RNN models, they are still setup to capture spatial features than temporal features. But the spatial features may have less weight when it comes to video as the movement is essential. Therefore ConvLSTMs are apt for this problem.

### 7.3 RESULTS FROM THE PREDICTIONS USING THE MODEL FOR IMPLEMENTATION

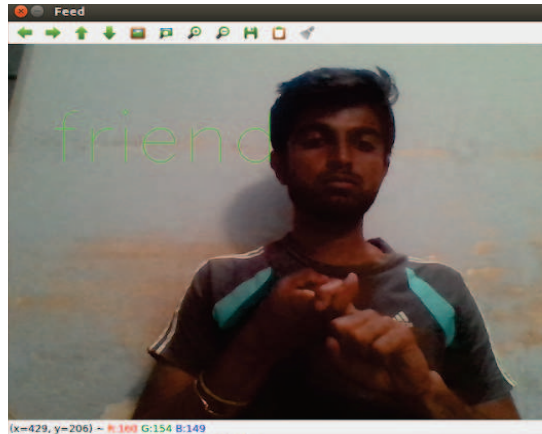
Real-time predictions are made by storing frames in a buffer, and once 80 frames are attained the resultant array of (80, 64, 64, 3) is passed onto the neural network model for prediction. The predicted class can be obtained by getting the element having the maximum value in the softmax probabilities array. This class prediction is outputted through the speaker, with the class representing the word being output.



**Fig. 7.8 (a). Prediction for “Hello”**



**Fig. 7.8 (b). Prediction for “Yes”**



**Fig. 7.8 (c). Prediction for “Friend”**

The output after each prediction is printed on the feed window for representation. This class prediction is converted into speech using the Python-Text-To-Speech library and outputted through the speaker connected to the Raspberry Pi. This process allows the sign language gestures to be converted into audio signals and thus enables a person carrying the portable device to communicate intelligibly with everyone.

## **CHAPTER 8**

### **CONCLUSION AND FUTURE WORK**

#### **8.1 CONCLUSION**

The proposed method for sign language recognition and translation is being developed using the sequential deep learning models with the help of Tensorflow, OpenCV, and audio conversion with the help of pyttsx3 library. The system was tested by performing sign language gestures in front of the camera and obtaining the input for processing by the neural network. The input is collected frame-by-frame, and they are stored in a buffer. After the collection of input in the required format, it is passed onto the neural network for making predictions. Once the prediction is made by the neural network, the predicted class is obtained from the softmax probabilities. This class is outputted in the form of audio signals through the speaker connected to the Raspberry Pi. After the word is outputted, the buffer is cleared to allow the storage of new frames representing the next word. This process is repeated to allow for translation of sign language in the form of gestures to speech, word-by-word. The limitations are identified and a future plan of action is created.

#### **8.2 FUTURE WORK**

The real-time processing speed could be improved by using more efficient algorithms and use of better processors. The method of capturing input could be varied to obtain much smoother experience in real-time applications and to reduce the delay after each word is translated.

The dataset created for the project contained a limited number of classes and a limited amount of data representing each class due to the resource constraints, the absence of which a larger dataset could be applied in the development process. The

dataset could be recorded in various environments by different people creating a diversity within the dataset and also allowing the model to focus only on the features corresponding to the gesture and not the features of the surrounding or the person performing the gesture.

A more accurate model could be trained by increasing the ConvLSTM layers and the number of filters and the dense layer neurons could also be increased, all of which could be applied on more powerful resources.

## REFERENCES

- [ 1] Van Hiep Phung, Eun Joo Rhee, “A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets”, *Applied Sciences Machine Learning Techniques Applied to Geospatial Big Data*, 2019.
- [ 2] Dongxu Li , Cristian Rodriguez Opazo, Xin Yu, Hongdong Li, “Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison”, *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.
- [ 3] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, Li Fei-Fei, “Large-scale Video Classification with Convolutional Neural Networks”, *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [ 4] Pascal Schneider, Raphael Memmesheimer, Ivanna Kramer, Dietrich Paulus, “Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping”, *Lecture Notes in Computer Science*, Vol. 11531., Springer Cham., 2019.
- [ 5] Hao Tang, Hong Liu, Wei Xiao, Nicu Sebe, “Fast and Robust Dynamic Hand Gesture Recognition via Key Frames Extraction and Feature Fusion”, *Neurocomputing*, Volume 331, 2019.
- [ 6] Dinh-Son Tran,Ngoc-Huynh Ho,Hyung-Jeong Yang,Eu-Tteum Baek,Soo-Hyung Kim, Guesang Lee, “Real-Time Hand Gesture Spotting and Recognition using RGB-D Camera and 3D-Convolutional Neural Network”, *2018 International Conference on Machine Learning and Machine Intelligence*, 2018.
- [ 7] Noorkholis Luthfil Hakim, Timothy K Shih, Sandeli Priyanwada Kasthuri Arachchi, Wisnu Aditya, Yi-Cheng Chen, Chih-Yang Lin, “Dynamic Hand Gesture

Recognition using 3D-CNN and LSTM with FSM Context-Aware Model”, Sensors (Basel), 2019.

[ 8] P. S. Neethu, R. Suguna, Divya Sathish, “An Efficient Method for Human Hand Gesture Detection and Recognition using Deep Learning Convolutional Neural Networks”, Springer Nature, 2020.

[ 9] Aditya Das, Shantanu Gawde, Khyati Suratwala, Dhananjay Kalbande, “Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images”, International Conference on Smart City and Emerging Technology, 2018.

[ 10] Brandon Garcia, Sigberto Alarcon Viesca, “Real-time American Sign Language Recognition with Convolutional Neural Networks”, 2016.

[ 11] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, Yunhao Liu, “Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges and Opportunities”, arXiv:2001.07416, 2020.

[ 12] Binjal Suthar, Bijal Gadhia, “Human Activity Recognition Using Deep Learning: A Survey”, Lecture Notes on Data Engineering and Communications Technologies, Volume 52, 2019.

[ 13] Negar Golestani, Mahta Moghaddam, “Human Activity Recognition using Magnetic Induction-based Motion Signals and Deep Recurrent Neural Networks”, Nature Communications, 2020.

[ 14] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, Wang-chun Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”, NIPS’15: Proceedings of the 28th International Conference on Neural Information, 2015.

[ 15] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[ 16] <https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7>

[ 17] <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>



## APPENDIX

### Source code for collecting of data of a particular class

```
import cv2
import os

if not os.path.exists('data/train/noaction'):
    os.makedirs('data/train/noaction')

if not os.path.exists('data/test/noaction'):
    os.makedirs('data/test/noaction')

vid = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'MP4V')

flag = 0
frame_count = 0
vid_count = len(os.listdir('data/train/noaction'))
path = 'data/train/noaction/'

while(True):
    ret, frame = vid.read()

    keyPress = cv2.waitKey(1) & 0xFF

    if keyPress == ord('s'):
        flag = 1
        out = cv2.VideoWriter(path + str(vid_count) + '.mp4', fourcc,
30, (640, 480))
```

```

        print("Writing")
    elif keyPress == ord('x'):
        fla = 0
        vid_count = vid_count + 1
        if vid_count == 50:
            path = 'data/test/hello/'
            out.release()
            print("Stopping")
    elif keyPress == ord('q'):
        break
    cv2.imshow("Feed", frame)

vid.release()
cv2.destroyAllWindows()

```

### **Source code for training of data**

```

import tensorflow as tf
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

classes = ["hello", "yes", "no", "friend", "noaction"]
img_height, img_width = 64, 64

def frame_collection(videoPath):
    frames = []
    vid = cv2.VideoCapture(videoPath)

```

```

count = 1

while count <= 80:
    ret, frame = vid.read()
    if ret:
        frame = cv2.resize(frame, (img_height, img_width))
        frames.append(frame)
        prev_frame = frame
    else:
        frames.append(prev_frame)
    count = count + 1
return frames

def dataset_extraction(inputdir):
    X = []
    Y = []

    classes_list = os.listdir(inputdir)

    for c in classes_list:
        print(c)
        files = os.listdir(os.path.join(inputdir, c))
        for f in files:
            print(f)
            frames =
frame_collection(os.path.join(os.path.join(inputdir, c), f))
            X.append(frames)
            y = [0] * len(classes)
            y[classes.index(c)] = 1
            Y.append(y)

```

```

X = np.asarray(X)
Y = np.asarray(Y)

return X, Y

train_data, train_class =
dataset_extraction("/content/drive/MyDrive/data/train")
test_data, test_class =
dataset_extraction("/content/drive/MyDrive/data/test")

model = tf.keras.models.Sequential([
    tf.keras.layers.ConvLSTM2D(64, (3, 3), return_sequences = True,
padding = "same", data_format = "channels_last", input_shape = (80,
img_height, img_width, 3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.ConvLSTM2D(8, (3, 3), return_sequences = False,
padding = "same", data_format = "channels_last"),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation = "relu"),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(5, activation = "softmax")
])

model.summary()
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
metrics = ['accuracy'])

```

```
history = model.fit(
    train_data,
    train_class,
    validation_data = (test_data, test_class),
    epochs = 5,
    batch_size = 8,
    shuffle = True
)

model.save('/content/drive/MyDrive/video_model1.model')

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

## Source code for classification of videos

```
import numpy as np
import cv2
import tensorflow as tf

vid = cv2.VideoCapture(0)
frame_count = 0
flag = 0
frames = []
classes = ["hello", "yes", "no", "friend", "no action"]
prediction = ""
model = tf.keras.models.load_model("video_model.model")

while True:
    ret, frameVid = vid.read()
    frame = cv2.resize(frameVid, (64, 64))

    if frame_count < 80 and flag == 1:
        frames.append(frame)
        frame_count = frame_count + 1
    elif frame_count == 80:
        frame_count = 0
        print("Done recording")
        modelFrames = np.array(frames)
        frames = []
        modelFrames = modelFrames.reshape((1, 80, 64, 64, 3))
        pred = model.predict(modelFrames)
        predictionNum = np.argmax(pred[0])
        prediction = classes[predictionNum]
        print(prediction)
```

```
        cv2.putText(frameVid, prediction, (50, 150),  
cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 255, 0))
```

```
cv2.imshow("Feed", frameVid)
```

```
vid.release()
```

```
cv2.destroyAllWindows()
```