**DESIGN AND DEVELOPMENT OF AN EMBEDDED SYSYTEM TO INTERRET SIGN LANGUAGE**

**A Project Report**

***Submitted By:***

**DEEAK RAJ SUBRAMANI – 2018504012**

**AKSHAYA SRIKANTH -2018504002**

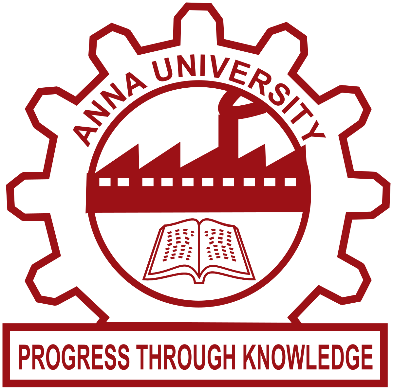
**MAHITA SELVRAJ -2018504554**

***In partial fulfilment for the award of the degree***

**of**

**BACHELOR OF ENGINEERING**

**in**



**ELECTRONICS AND COMMUNICATION**

**DEPARTMENT OF ELECTRONICS ENGINEERING**

**MADRAS INSTITUTE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600044**

**JUNE 2022**

**ACKNOWLEDGEMENT**

We consider it as a privilege and our primary duty to express our gratitude and respect to all those who guided and inspired us in the successful completion of the project.

We owe solemn gratitude to **Dr J. PRAKASH**, Dean, Madras Institute of Technology, for having given consent to carry out the project work at MIT Campus, Anna University.

We wish to express our sincere appreciation and gratitude to **Dr. M. GANESH MADHAN,** Professor and Head of the Department of Electronics Engineering, who has encouraged and motivated us in our endeavors.

We are extremely grateful to our project guide **Dr.V. SATHIESH KUMAR**, Teaching Fellow, Department of Electronics Engineering, for her timely and thoughtful guidance and encouragement for the completion of the project.

We sincerely thank all our panel members **Dr. S. P. JOY VASANTHA RANI, Dr. A. VIJI, and MR. N. JAGADEESH KUMAR** for their valuable suggestions. We also thank all the teaching and non-teaching staff members of the Department of Electronics Engineering for their support in all aspects.

**DEEAK RAJ SUBRAMANI – 2018504012**

**AKSHAYA SRIKANTH -2018504002**

**MAHITA SELVRAJ -2018504554**

**BONAFIDE CERTIFICATE**

Certified that this Project Report titled **“Design and Development of an Embedded System to Interpret Sign Language”** is the bonafide work of **DEEAK RAJ SUBRAMANI (2018504012), AKSHAYA SRIKANTH (2018504002) & MAHITA SELVRAJ (2018504554)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Dr. M. GANESH MADHAN**

**Professor and Head**

Department of Electronics Engineering

Madras Institute of Technology

Anna University

Chennai - 600044

**Dr.V. SATHIESH KUMAR**

**Assistant Professor and Guide**

Department of Electronics Engineering

Madras Institute of Technology

Anna University

Chennai - 600044

**ABSTRACT**

According to the census of 2011, almost 1.65 percentage of the sort of total disable population for the most part have speaking disability and around 4.1 percent for all have hearing disability. Over the years, generally several communication systems mostly were developed throughout the world, including very ancient methods like print on palm. Gradually using an interpreter became quite fairly common for the disable people to communicate fairly more efficiently. Although these methods specifically were present, people mostly found these methods depending and laborious. This led to the need for developing sign language systems for their efficient communication. The main aim of this project is to make the communication for the disabled population (deaf and dumb) more efficient, which basically is being independent. They should be able to essentially communicate without depending on others like using an interpreter. Hence, we acknowledged the need to develop an embedded system that interprets sign language. We designed and developed an embedded system that overcomes the difficulties faced by deaf and dumb. To make it more useful in regular lives, we have also introduced an audio output at the end that reads the interpreted word in English language.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **5** |
|  | **LIST OF TABLES** | **9** |
|  | **LIST OF FIGURES** | **10** |
|  | **LIST OF ABBREVIATIONS** | **12** |
| **1** | **INTRODUCTION** | **13** |
|  | * 1. OVERVIEW   2. CONVOLUTIONAL NEURAL NETWORK   3. ARCHITECTURE OF CNN   1.3.1 CONVOLUTIONAL LAYER  1.3.2 POOLING LAYER  1.3.3 FLATTEN LAYER  1.3.4 FULLY CONNECTED LAYER  1.3.5 DENSE LAYER   * 1. ARCHITECTURE OF RNN   2. RASPBERRY PI | 13  13  14  15  16  17  18  18  18  19 |
| **2** | **LITERATURE SURVEY** | **21** |
| **3** | **OVERVIEW OF THE PROJECT** | **24** |
|  | 3.1 SOFTWARE IMPLEMENTATION  3.2 FLOWCHART FOR SOFTWARE IMLEMENTATION  3.3 EXPLANATION FOR THE FLOWCHART  3.4 DATASET USED  3.5 FRAME EXTRACTION AND PREPROCESSING  3.5.1 FRAMES EXTRACTED  3.5.2 DATA PREPROCESSING  3.5.2.1 Image Resizing  3.5.2.2 Remove Background  3.6 TRAINING ARCHITECTURE  3.7 OPTIMIZERS USED  3.7.1 SGD  3.7.2 ADADELTA  3.7.3 ADAMAX  3.7.3.1 Learning Rate  3.7.3.2 Batch Size  3.8 HARDWARE IMPLEMENTATION  3.8.1 COMPONENTS USED  3.8.2 FLOWCHART FOR HARDWARE IMPLEMENTATION  3.8.3 EXPLANATION FOR THE FLOWCHART | 24  24  25  25  27  27  28  28  28  28  29  29  31  32  32  33  33  33  34  34 |
| **4** | **RESULTS AND DISCUSSIONS** | **35** |
|  | 4.1 COMPARISION OF OPTIMIZERS  4.1.1 PREDICTION RESULTS  4.2 VARIATION OF LEARNING RATE  4.3 VARITION OF BATCH SIZE  4.4 CONFUSION MATRIX | 35  35  36  36  37 |
| **5** | **CONCLUSION AND FUTURE WORK** | **38** |
| **6** | **REFERENCES** | **39** |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| 3.1 | NUMBER OF VIDEOS AND FRAMES IN EACH CLASS | 25 |
| 3.2 | COMONENTS USED FOR HARDWARE IMPLEMENTATION | 33 |
| 4.1 | COMPARISON OF DIFFERENT OPTIMIZER | 35 |
| 4.2 | DIFFERENT LEARNING RATE USED | 36 |
| 4.3 | DIFFERENT BATCH SIZE USED | 36 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 1.1 | ARCHITECTURE OF CNN | 14 |
| 1.2 | CONVOLUTION OPERATION | 15 |
| 1.3 | STRIDING OPERATION | 16 |
| 1.4 | ZERO PADDING | 16 |
| 1.5 | POOLING OPERATION | 17 |
| 1.6 | FULLY CONNECTED LAYER | 18 |
| 1.7 | ARCHITECTURE OF RNN | 19 |
| 1.8 | RASPBERRY PI | 20 |
| 2.1 | BLOCK DIAGRAM OF HAND GESTURE RECOGNITION SYSTEM FOR LSA | 22 |
| 3.1 | BLOCK DIAGRAM OF SOFTWARE IMPLEMENTATION | 24 |
| 3.2-3.11 | FRAMES EXTRACTED | 27 |
| 3.12 | TRAINING ARCHITECTURE | 28 |
| 3.13-3.14 | MODEL LOSS AND MODEL ACCURACY FOR SGD | 30 |
| 3.15-3.16 | MODEL LOSS AND MODEL ACCURACY FOR ADADELTA | 31 |
| 3.17-3.18 | MODEL LOSS AND MODEL ACCURACY FOR ADADELTA | 32 |
| 3.19 | BLOCK DIAGRAM OF HARDWARE IMPLEMENTATION | 34 |
| 4.1 | HEATMAP OF CONFUSION MATRIX | 37 |

**LIST OF ABBREIVIATIONS**

|  |  |
| --- | --- |
| **TERM** | **EXPLANATION** |
| ISL | Indian Sign Language |
| ASL | American sign language |
| BANZSL | British, Australian and New Zealand Sign Language |
| CSL | Chinese Sign language |
| LSF | French Sign Language |
| CNN | Convolutional Neural Networks |
| 2D | 2-Dimensional |
| 1D | 1-Dimensional |
| RNN | Recurrent Neural Networks |
| ARM | Advanced RISC Machines |
| RISC | Reduced Instruction Set Computer |
| USB | Universal Serial Bus |
| USB-C | Universal Serial Bus Type-C |
| HDMI | High-Definition Multimedia Interface |
| SVM | Support Vector Machines |
| OH | Orientation Histogram |
| PCA | Principal Component Analysis |
| IIIT-A | Indian Institute of Information Technology, Allahabad |
| EOS | Electro-Optical System |
| VGG16 | Visual Geometry Group |
| SGD | Stochastic Gradient Descent |
| LR | Learning Rate |

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Sign language as the word implies, involves using a visual means of communication by showing hand signals, gestures, facial expression, and body expression. This represents the main form of communication for the deaf and hard of hearing. Sign language is not universal, which means that there is no single sign language that is being used all over the world. Around 138-300 different types of sign languages are being used all over the globe today. An interesting fact includes that most countries that share the same spoken language does not share the same sign language.

A sign language journey is usually started by learning the A-Z or alphabetical equivalent in sign language form. Fingerspelling denotes the use of hands to represent individual letters of a written alphabet. The different types of sign languages that are used all over the world include Indian Sign Language (ISL), American sign language (ASL), British, Australian and New Zealand Sign Language (BANZSL), Chinese Sign language (CSL), French Sign Language (LSF) and many more.

**1.2 CONVOLUTIONAL NEURAL NETWORK**

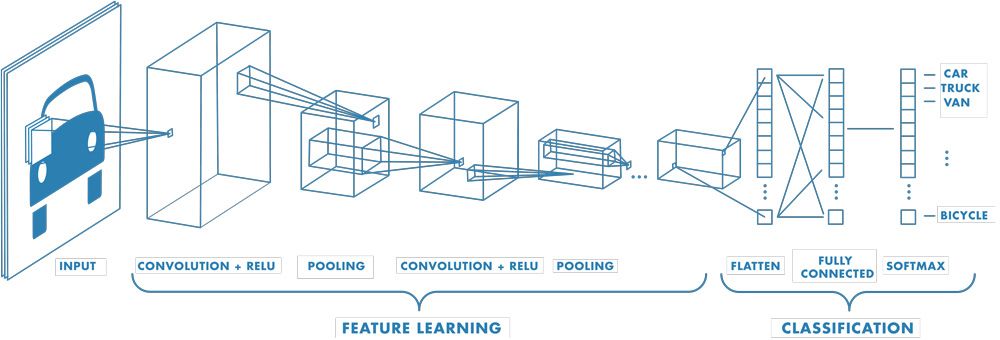
CNN is a fusion of convolutional Layers and Neural Networks. It is a type of [artificial neural network](https://www.techtarget.com/searchenterpriseai/definition/neural-network) used in [image recognition](https://www.techtarget.com/searchenterpriseai/definition/image-recognition) and processing that is specifically designed to process pixel data. It mainly consists of following layers.

* Input Layer
* Convolutional Layer
* Pooling Layer and
* Flatten Layer

**1.3ARCHITECTURE OF CNN**

The architecture of CNN model is shown in Figure 1 and there five different layers in CNN.

* Convolutional Layer
* Pooling Layer
* Fully connected Layer
* Flatten Layer
* Dense Layer



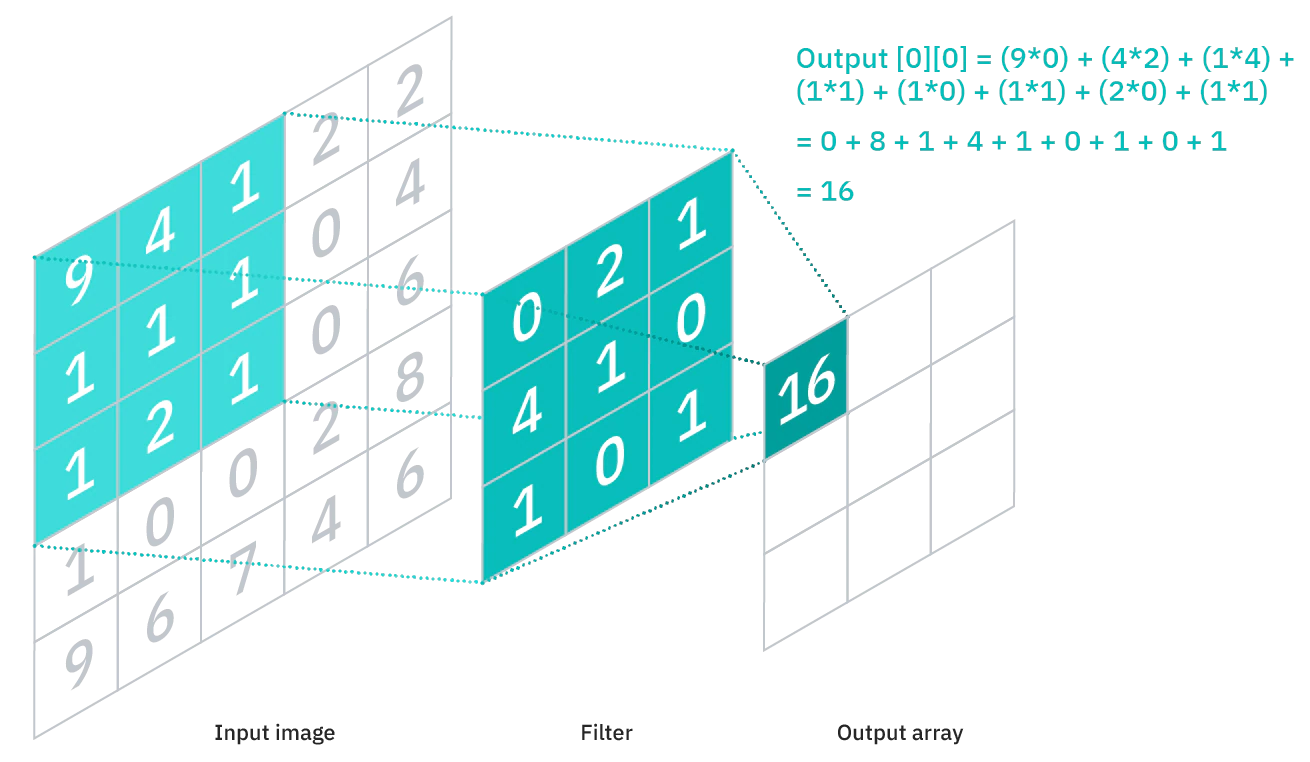
**Figure 1.1 Architecture of CNN**

**1.3.1CONVOLUTIONAL LAYER**

A convolutional layer is the main building block of a CNN. This layer is responsible for feature extraction. This layer consists of three major operations:

1. Convolution
2. Stride
3. Padding
4. **Convolution**

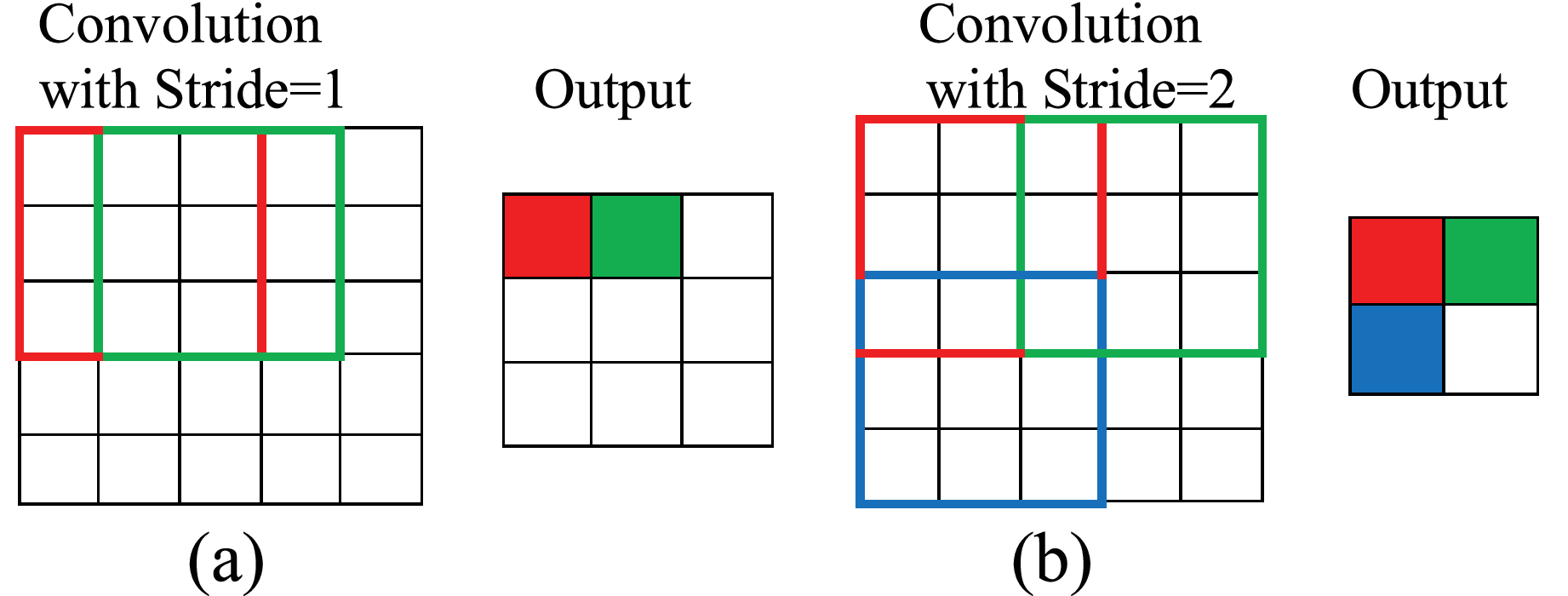
Convolution contains a set of filters (or kernels), parameters of which are to be learned throughout the training. It is basically a matrix multiplication between the image matrix and a filter or kernel.



**Figure 1.2 Convolution Operation**

1. **Stride**

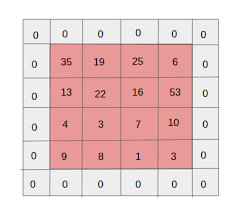
Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. If the stride is 2 then the kernel is moved 2 pixels at a time and once the strides increase the pixels moved is also increased.



**Figure 1.3 Striding Operation**

1. **Padding**

Padding is simply a process of adding layers of zero to our input images. This is done in order to prevent the image being shrunk.

****

**Figure 1.4 Zero Padding**

**1.3.2 POOLING LAYER**

The Pooling layer, like the Convolutional Layer, is in charge of shrinking the Convolved Feature's spatial size. Through dimensionality reduction, this will reduce the amount of computing power needed to process the data. Furthermore, it aids in properly training the model by allowing the extraction of dominating characteristics that are rotational and positional invariant.

Max Pooling and Average Pooling are the two different forms of pooling. The largest value from the area of the picture that the Kernel has covered is returned by Max Pooling. The average of all the values from the area of the picture covered by the Kernel is what is returned by average pooling, on the other hand.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.



**Figure 1.5 Pooling Operation**

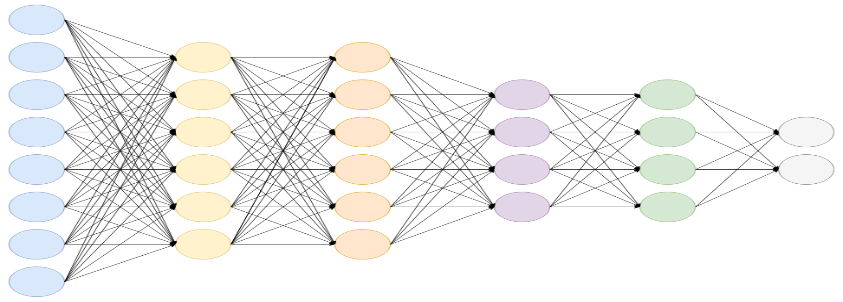
**1.3.3 FLATTEN LAYER**

Flatten is used to flatten the input. It usually converts 2D image into 1D. For example, if flatten is applied to layer having input shape as (batch\_size, 2,2), then the output shape of the layer will be (batch\_size, 4).

**1.3.4 FULLY CONNECTED LAYER**

Fully Connected Layer is simply,[feed forward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network). Fully Connected Layers form the last few layers in the network. It is used to connect it to all layers.

The **input** to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattenedand then fed into the fully connected layer.



**Figure 1.6 Fully Connected Layer**

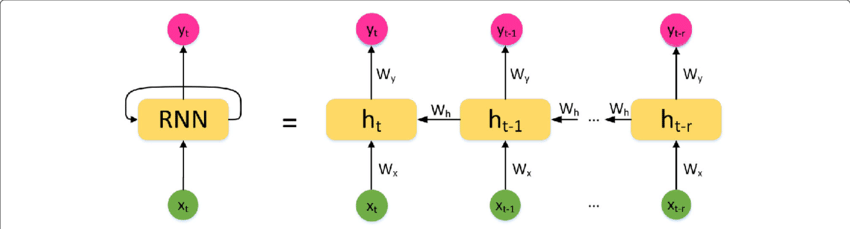
**1.3.5 DENSE LAYER**

A layer that is densely linked to the layer above it is one in which every neuron in the layer is coupled to every other neuron in the layer above. In artificial neural network networks, this layer is the one that is most frequently utilised.

**1.4 ARCHITECTURE OF RNN**

RNNs are a sort of neural network in which the output from one phase is used as the input for the next. Traditional neural networks have inputs and outputs that are independent of one another, but in situations when it is necessary to anticipate the next word in a phrase, it is necessary to remember the prior words. As a result, RNN was developed, which utilized a Hidden Layer to resolve this problem. The hidden state of an RNN, which retains some information about a sequence, is its primary and most significant characteristic.

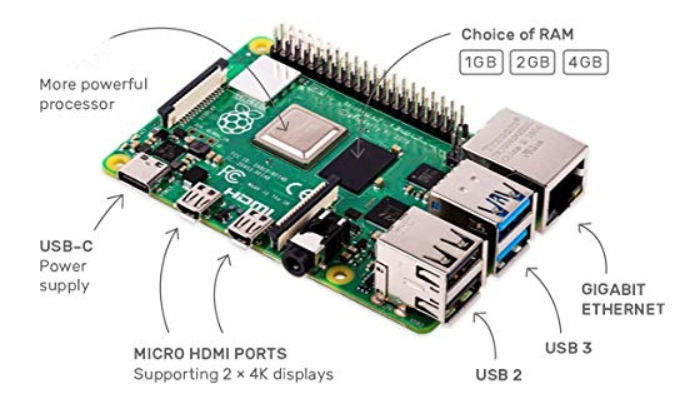
RNNs have a "memory" that retains all data related to calculations. It executes the same action on all of the inputs or hidden layers to generate the output, using the same settings for each input. Unlike other neural networks, this one has less complicated parameters, unlike other neural networks.



**Figure 1.7 Architecture of RNN**

**1.5 RASPBERRY PI**

The processor used is, quad-core Cortex-A72 (ARM v8). The memory capacity can 1 GB, 2 GB, 4 GB, 8 GB. It consists of Ethernet port, USB ports, micro-HDMI ports and USB-C port. There are 40 General Purpose Input Output pins.

****

**Figure 1.8 Raspberry Pi**

**CHAPTER 2**

**LITERATURE SURVEY**

1.Computer cameras are used as the input device for hands or finger information in vision-based approaches. The Vision Based approaches just need a camera, enabling a seamless contact between people and computers without the need for any additional hardware. By defining artificial vision systems that are implemented in software and/or hardware, these systems frequently supplement biological vision. This is a difficult issue since in order for these systems to execute in real time, they must be backdrop invariant, lighting insensitive, human and camera independent. Additionally, these systems need to be adjusted to satisfy the demands, which include accuracy and resilience.

Although vision-based analysis is based on how people perceive information about their environment, it is likely the hardest to perform well. So far, a number of various strategies have been tried.

One option is building a three-dimensional model of the human hand. One or more cameras match the model to photos of the hand, and joint angles and characteristics pertaining to palm orientation are computed. The categorization of gestures is then done using these factors.

The second step is to take the image using a camera, then extract certain features, and then utilize those features as input into a classification system.

2. The Argentinian sign language (LSA) hand gesture identification method is put forth in this research. The establishment of a database of handshapes for the Argentinian Sign Language is the first of this paper's two significant accomplishments. Secondly, a supervised adaption of self-organizing maps called ProbSom. It is a probabilistic adaptation of Kohonen’s self-organizing maps used for image processing, descriptor extraction, and handshape classification. Other state-of-the-art methods are contrasted with this one, including Support Vector Machines (SVM), Random Forests, and Neural Networks. Utilizing the suggested descriptor, the ProbSom-based neural classifier had an accuracy rate of more than 90%.

Extract Descriptors Using SIFT and Radon

Classification using Probsom model

Preprocessing

**Figure 2.1 Block Diagram of Hand Gesture Recognition System for LSA**

3.Data acquisition, preprocessing, feature extraction, and classification are the four main elements that make up the suggested system. Skin Filtering and histogram matching are preprocessing steps that are followed by Eigen­vector based Feature Extraction and an Eigen value weighted Euclidean distance based Classification Technique. In this work, a 96 percent identification rate was achieved using 24 distinct alphabets.

4.It is a very difficult research problem to distinguish continuous motions from sign language movements. The key frame extraction approach based on gradients was used by the researchers to tackle this issue. These key frames were useful for eliminating unnecessary frames and breaking up continuous sign language motions into sequences of signals. Each gesture has been addressed separately following the division of gestures. Then, using the Orientation Histogram (OH) and Principal Component Analysis (PCA) to reduce the dimension of the features generated after the OH, features of the pre-processed gestures were retrieved. In the Robotics and Artificial Intelligence Laboratory (IIIT­A), experiments were run on their own continuous ISL dataset that was produced using a Canon EOS camera. Different types of classifiers, such as Euclidean distance, correlation, Manhattan distance, city block distance, etc., were used to test the probes. With several distance classifier types, comparative examination of their suggested scheme was done. They discovered from the investigation above that results from correlation and Euclidean distance provide findings with more accuracy than other classifiers.

5. In this study, statistical methods for simultaneous identification of ISL movements, including both hands, are illustrated. The authors built and used a video library that included a variety of movies for many different signs. Due of its popularity for illumination and orientation invariance, the direction histogram is the feature utilized for classification. Euclidean distance and K-nearest neighbour metrics were two separate methods used for recognition.

**CHAPTER 3**

**OVERVIEW OF PROJECT**

It is proposed to design an embedded system that can interpret sign language. This is done by using the concept of CNN and RNN. This will be done in both software and hardware.

**3.1 SOFTWARE IMPLEMENTATION**

There are many architectures which can be used on CNN model. In this work VGG16 architecture is used. The entire coding is done in Python Language.

**3.2 FLOWCHART FOR SOFTWARE IMPLEMENTATION**

Pool Layer

Convolutional layer

Import all libraries

Frame extraction and preprocessing

Train RNN

Testing

**Figure 3.1 Block Diagram of Software Implementation**

The above figure shows the process undergone to predict the sign language.

**3.3 EXPLANATION FOR THE FLOWCHART**

**Step 1:** First we need to import all the required libraries.

**Step 2:** The input is reduced into frames and preprocessed

**Step 3:** The frames then is passed onto the convolutional layer for feature extraction.

**Step 4:** It is then passed to pool layer

**Step 5:** Now it is given to RNN for sequential output

**Step 6:** The final testing is done

Initially all the necessary package is installed. The input is video dataset

**3.4 DATASET USED**

**Table 3.1 Number of videos and frames in each class**

|  |  |  |  |
| --- | --- | --- | --- |
| **CLASSES** | **NO OF VIDEOS** | **NO OF FRAMES PER VIDEO** | **FRAME SIZE** |
| Alive | 8 | 10 | 576X324 |
| Bad | 8 | 10 | 576X324 |
| Beautiful | 8 | 10 | 576X324 |
| Big large | 8 | 10 | 576X324 |
| Blind | 8 | 10 | 576X324 |
| Cheap | 8 | 10 | 576X324 |
| Clean | 8 | 10 | 576X324 |
| Cool | 8 | 10 | 576X324 |
| Dead | 8 | 10 | 576X324 |
| Fast | 8 | 10 | 576X324 |
| Happy | 8 | 10 | 576X324 |
| Hot | 8 | 10 | 576X324 |
| Light | 8 | 10 | 576X324 |
| Poor | 8 | 10 | 576X324 |
| Sad | 8 | 10 | 576X324 |
| Tall | 8 | 10 | 576X324 |
| Warm | 8 | 10 | 576X324 |
| Wide | 8 | 10 | 576X324 |
| Young | 8 | 10 | 576X324 |

There are 20 classes in which each class comprises 8 videos.

**3.5 FRAME EXTRACTION AND PREPROCESSING**

From the 20 datasets each video is converted into 10 frames each with a resolution of 576 x 324.

**3.5.1 FRAMES EXTRACTED**









**Figure 3.2-3.11 Frames Extracted**

**3.5.2 DATA PREPROCESSING**

Data preprocessing is a very important step. Image Resizing and Remove Background has been performed before training and testing the model.

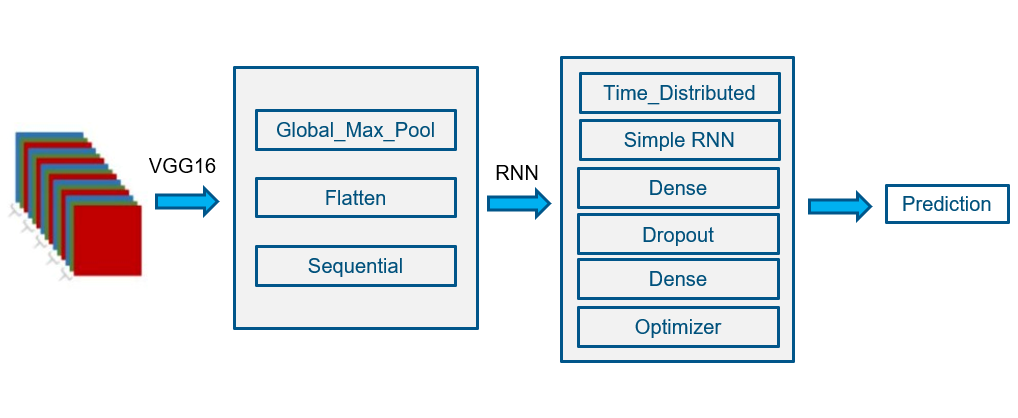
**3.5.2.1 Image Resizing**

Dataset consists of Sign Language of various dimensions. All the Frames are resized to 576X324.

**3.5.2.2 Remove Background**

The right and left background apart from our hand gesture and body is removed in order to train the model.

**3.6 TRAINING ARCHITECTURE**

****

**Figure 3.12 Training Architecture**

The VGG16 model is a variation of the VGG model that has 16 convolutional layers. The architecture is uniform. There are 5 blocks in all. The most distinctive feature of VGG16 is that it prioritised having convolution layers of a 3x3 filter with a stride 1 and always utilised the same padding and maxpool layer of a 2x2 filter with a stride 2. Throughout the whole design, convolution and max pool layers are arranged in the same manner. The 16 in VGG16 denotes the fact that there are 16 layers with weights. There are around 138 million parameters in this network, making it a sizable network. After going through this, we get more than three channel. This global max pool layer receives the output from this model and reduces the dimensionality. The output is sent to a flattening layer, which flattens the output for improved vector representation. Sequential data are trained using the provided sequential model. The distributed layer receives the output of the VGG16 and delivers 10 timesteps sequentially to the RNN. As previously said, RNN employs sequential data, i.e., history, to aid with video prediction.

Similar to a completely linked layer is a dense layer. It is connected to all levels using it. To prevent overfitting in the model, a dropout layer is utilised. Once more, a dense layer is utilised as a completely linked layer. To decrease loss and increase accuracy, findings are optimised using an optimizer.

**3.7 OPTIMIZERS USED**

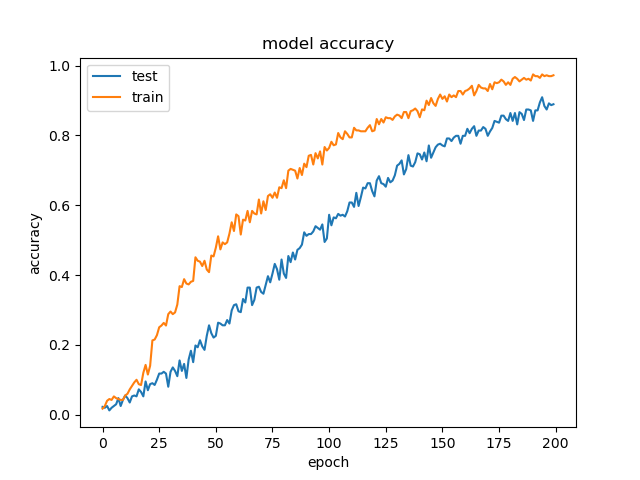
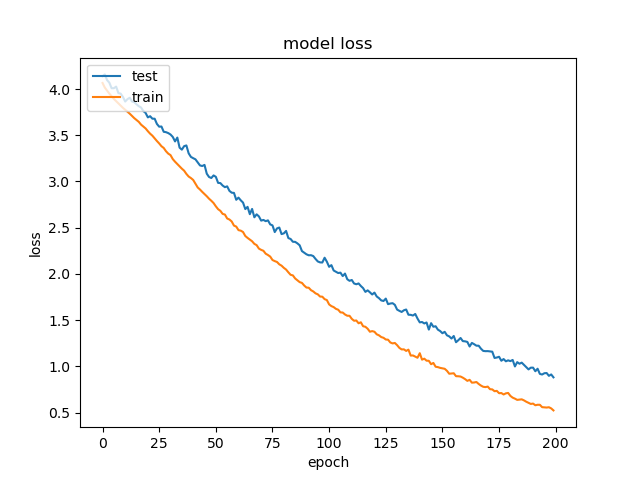
**3.7.1 SGD**

In SGD, each iteration is carried out with just one sample, or a batch size of one. For the iteration, the sample is chosen and randomly shuffled.

Therefore, in SGD, rather of computing the gradient of the cost function for all instances as a whole, we compute the gradient for a single example's cost function at each iteration.

The path travelled by the algorithm to reach the minima in SGD is typically noisier than the one taken by a standard Gradient Descent algorithm since just one sample from the dataset is randomly selected for each iteration. But as long as we get at the minima and with a noticeably lower training period, it doesn't really matter which path the algorithm takes.

One thing to keep in mind is that because SGD tends to be more random in its descent than normal Gradient Descent, it often required more rounds to achieve the minima. Although it takes more iterations than usual Gradient Descent to reach the minima, it is still computationally considerably less expensive than usual Gradient Descent. Consequently, SGD is recommended over Batch Gradient Descent in the majority of situations for improving a learning algorithm. This optimizer is used in the model it’s accuracy and loss is calculated and plotted below.



**Figure** **3.13-3.14 Model Loss and Model Accuracy for SGD**

**3.7.2 ADADELTA**

Adadelta takes the effective rate of previous steps to the current gradient, therefore we don't need to define a default reading rate.

With the Adagrad algorithm, there are three main issues.

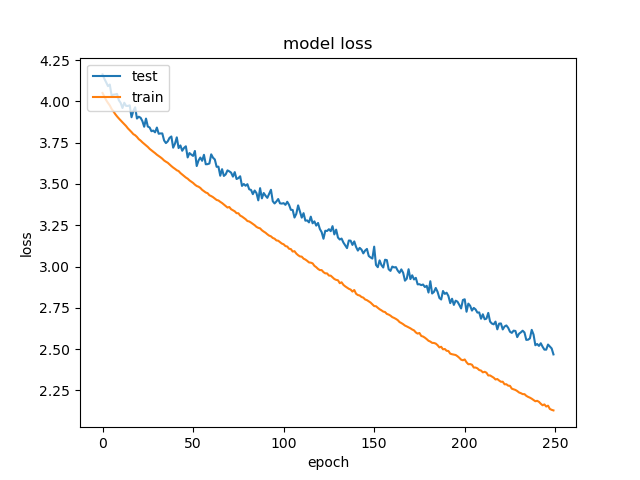
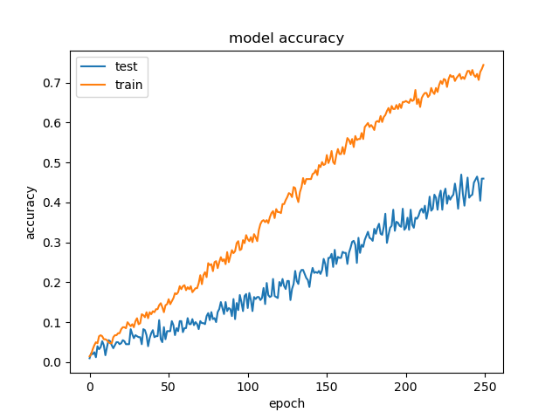
1)The pace of learning is monotonically declining.

2)The late training phase has a relatively low learning rate.

3)The first global learning rate must be performed in order to set it.

Adadelta is an extension of Adagrad that likewise makes excessive attempts to slow down Adagrad's rate of learning. It accomplishes this by setting a size limit on the gradient window that has been surpassed. So, the running average at time is dependent on both the current gradient and the prior average. In Adadelta, we take the ratio of the running average of the past time steps to the current gradient, therefore we don't need to select the default learning rate.

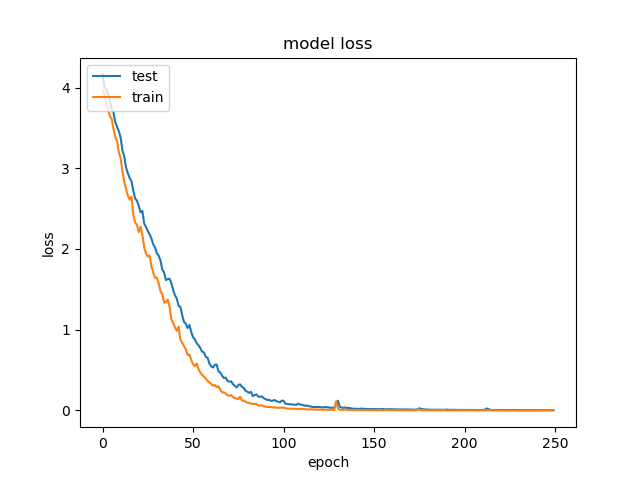
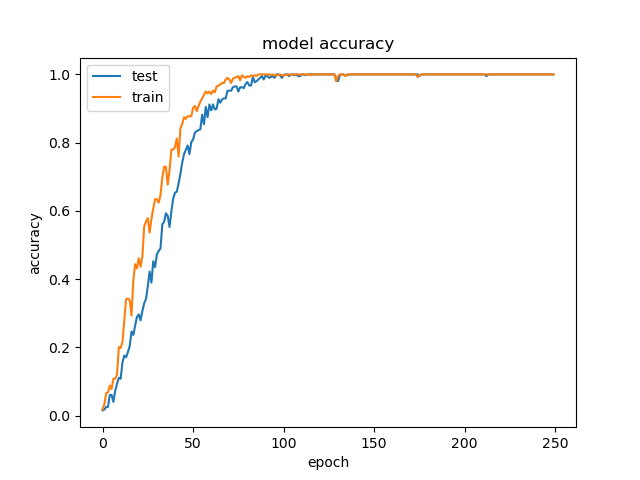
Now we changed the optimizer and acquired the graph.

**Figure 3.15-3.16 Model Loss and Model Accuracy for Adadelta**

**3.7.3 ADAMAX**

AdaMax is an alteration of the Adam optimizer. It is based on low-order adaptive approximation moments (based off on infinity norm). Sometimes in the case of embeddings, AdaMax is considered better than Adam. Also, Infinite order makes the algorithm stable. Requires less tuning on hyperparameters

** **

**Figure 3.17-3.18 Model Loss and Model Accuracy for Adamax**

Now we vary the learning Rate and Batch Size by fixing the optimizer.

**3.7.3.1 Learning Rate**

How quickly the neural network would converge to minima depends on its

value. To gain the best value for LR, we typically pick a learning rate and

modify it based on the outcomes. The process of convergence would be

exceedingly sluggish if the learning rate were too low for the neural network,

and quickly if it were too high. Learning rate is set to 0.1,0.01,0.0001,0.00005

**3.7.3.2 Batch Size**

the batch size is the number of samples that will be passed through to the network at one time. The batch size is the number of samples that are passed to the network at once. Here we have changed the batch size as 20, 50,100.

**3.8 HARDWARE IMPLEMENTATION**

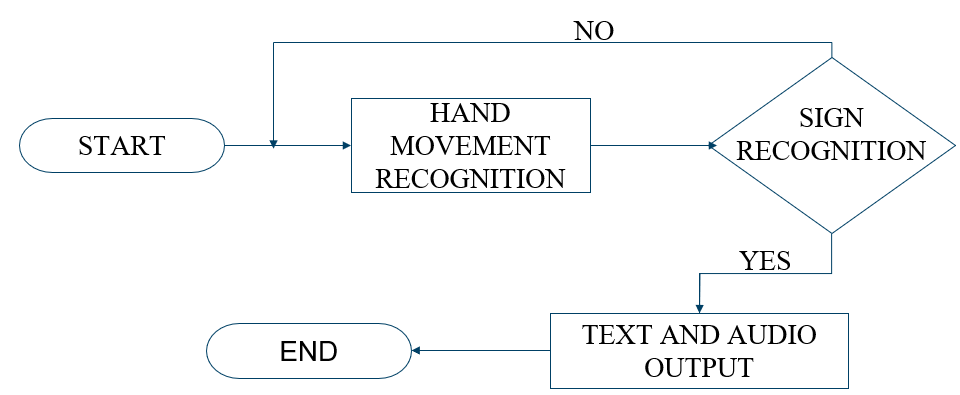
The trained model is now passed on to the hardware for execution of the final results

**3.8.1 COMPONENTS USED**

**Table 3.2 Components used for hardware implementation**

|  |  |  |
| --- | --- | --- |
| **COMPONENTS USED** | **SPECIFICATION** | **NUMBER USED** |
| Raspberry Pi 4 | **-** | 1 |
| LCD Screen | 155mm x 86mm | 1 |
| Web Camera |  | 1 |

**3.8.2 FLOWCHART FOR HARDWARE IMPLEMENTATION**

****

**Figure 3.19 Block Diagram of Hardware Implementation**

**3.8.3 EXPLANATION FOR THE FLOWCHART**

**Step 1:** First the web camera is turned on.

**Step 2:** The hand movement is first recognized.

**Step 3:** If sign language is recognized the process continues or else it again goes to hand movement recognition.

**Step 4:** The sign language is then passed on to the model.

**Step 5:** Then the output is given in text and audio format.

**CHAPTER 4**

**RESULTS AND DISCUSSIONS**

Testing dataset consists of 128 videos. These videos are preprocessed with Image Resizing and Remove background.

**4.1COMPARISON OF OPTIMISERS**

**Table 4.1 Comparison of different optimiser**



**4.1.1 PREDICTION RESULTS**

Now from the above table adamax optimizer has better accuracy. Now we need to change the learning rate and batch size in order to find a better accuracy. This is done by fixing the optimizer i.e., AdaMax.

**4.2 VARIATION OF LEARNING RATE**

**Table 4.2 Different Learning Rate used**

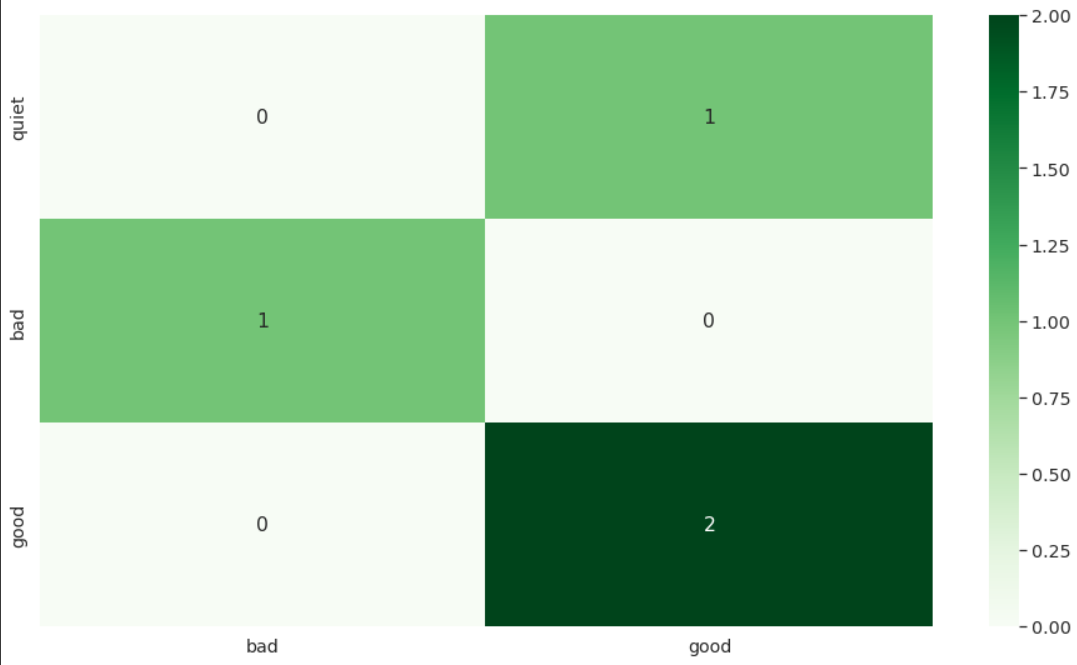
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Learning Rate | Epoch Reached | Accuracy  (%) | Loss | Val\_accuracy  (%) | Val\_loss |
| 0.1 | 12 | 6.25 | 2.79 | 6.25 | 2.77 |
| 0.01 | 1 | 6.25 | 5.063 | 6.25 | 5.88 |
| 0.001 | 23 | 6.25 | 2.81 | 12.5 | 2.77 |
| 0.0001 | 45 | 98.43 | 0.023 | 97.66 | 0.029 |
| **0.00005** | **63** | **99.2** | **0.0016** | **99.34** | **0.00041** |

**4.3 VARIATION OF BATCH SIZE**

**Table 4.3 Different Batch Size used**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Batch size | Epoch Reached | Accuracy  (%) | Loss | Val\_accuracy  (%) | Val\_loss |
| 10 | 42 | 96.10 | 0.207 | 96.31 | 2.77 |
| 20 | 63 | 99.2 | 0.0016 | 99.34 | 0.00041 |
| 50 | 134 | 99.34 | 0.0013 | 99.47 | 0.00034 |
| **100** | **291** | **99.45** | **0.00059** | **99.6** | **0.00017** |

**4.4 CONFUSION MATRIX**

****

**Figure 4.1 Heatmap of confusion matrix**

**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

Thus, an embedded system for interpreting sign language has been designed and implemented in both hardware and software. In software, we used a web camera to capture the sign language in real-time. Depending on the training, the real-time sign language is detected. The detected sign language is also converted and read out in audio format. In the hardware part, the corresponding word for the detected sign language is displayed on lcd screen. Hence it can be said that the model can be used in real-time applications. In the future it is proposed to increase the number of classes of datasets and also the corresponding outputs in different languages to widen the usage of this sign language interpreter system.

**REFRENCES**

1. Ronchetti, Franco, Facundo Quiroga, César Armando Estrebou, and Laura Cristina Lanzarini (2016). "Hand Shape recognition for Argentinian sign language using probsom." Journal of Computer Science & Technology 16.
2. Singha, Joyeeta, and Karen Das, (2015). "Automatic Indian Sign Language Recognition for Continuous Video Sequence." ADBU Journal of Engineering Technology 2, no. 1.
3. Tripathi, Kumud, and Neha Baranwal GC Nandi (2015). "Continuous Indian Sign Language Gesture Recognition and Sentence Formation." Procedia Computer Science 54 :523­531.
4. Nandy, Anup, Jay Shankar Prasad, Soumik Mondal, Pavan Chakraborty, and Gora Chand Nandi. (2010) "Recognition of isolated Indian sign language gesture in real time." Information Processing and Management:102­107.
5. Real-Time Sign Language Gesture (Word) Recognition from Video Sequences Using CNN and RNN Sarfaraz Masood, Adhyan Srivastava, Harish Chandra Thuwal and Musheer Ahmad
6. Pigou, L., Dieleman, S., Kindermans, P.-J., Schrauwen, B. (2014), “Sign language recognition using convolutional neural networks”. Workshop at the European Conference on Computer Vision 2014. Springer International Publishing, pp. 572–578.
7. Sharma, R., Bhateja, V., Satapathy, S.C., Gupta, (2017) “Communication device for differently abled people: a prototype model”. Proceedings of the International Conference on Data Engineering and Communication Technology. Springer, Singapore, pp. 565–575.
8. Masood, S., Thuwal, H.C., Srivastava, (2018) “American sign language character recognition using convolution neural network”. Proceedings of Smart Computing and Informatics, Springer, Singapore pp. 403–412.