Realization and Implementation of Communication for Hearing Impaired and Inarticulate People

September 23, 2021

Department of Electrical Engineering

(Group Members)

Muhammad Faizan Ikram	**2018-UET-NML-ELECT-27**
Tanzila Iram	**2018-UET-NML-ELECT-34**

(Supervisor)

Dr. Sajjad Ur Rehman
Department of Electrical Engineering

(Co-Supervisor)
Dr. Majid Ali
Department of Electrical Engineering

Namal College, Mianwali
Affiliated with
University of Engineering and Technology, Lahore

Table of Contents

List of Figures

List of Tables

Abstract

Natural language processing is one of the most growing fields of research. It deals with human-computer interaction and natural languages. Humans share ideas and thoughts with people around them using speech and hearing abilities. But this is not the case for hearing impaired and inarticulate people. Through sign recognition, communication is possible with hearing impaired and inarticulate people. This project aims to develop a mobile application-based system for recognizing sign language and converting it into text and speech, which provides smooth communication between the deaf-mute community and normal people, thereby reducing the communication barrier between them. Our proposed system will consist of a mobile application that would be used for communication. The project will mainly consist of two parts, i.e., sign-to-speech conversion and mobile application development. In sign-to-speech conversion, the sign will be recognized using a mobile camera with the help of computer vision, image processing and machine learning algorithms. The recognized sign will be converted into text, and text will be used to generate speech. With the incorporation of the mobile app, this system will ease people with disabilities to communicate with non-disabled people and reduce the communication barrier between them.

Chapter 1

1 Introduction

Natural Language Processing (NLP) is an emerging field that helps in conveying information and meaning with semantic cues such as words, signs, or images. Sign Language is a very convenient way for hearing impaired and inarticulate people. But for other world, communication with hearing impaired and inarticulate people is difficult because a normal person cannot understand their sign language in normal circumstances. This difference in society can be minimized if we can program a machine in such a way that it translates sign language into text or speech. This project utilized computer vision and image processing techniques to develop a system that can convert sign language into textual and audio form with a mobile application interface, thereby, reducing the communication barrier between specially-abled and non-disabled people. It is a reliable alternate mode of communication that does not need a voice and is critical for individuals suffering from hearing and speech disabilities to improve their quality of life. This project aims to meet UN's SDG 8.5, 10.2 and 10.3. It aims to increase the productive employment of all persons including persons with disabilities. The second purpose of this project is to reduce inequalities and communication barriers in society.

The sign language of hearing-impaired and inarticulate people will be recorded in the data collection phase in the form of videos consisting of words and sentences that are used in daily life and emergencies. Since video is dynamic in nature and it is a collection of images that are referred to as frames, therefore it is not much different from an image classification problem. These frames will be processed in sequence and will be used to extract features with the help of deep learning algorithms, and then classify those frames based on these extracted features.

A video consists of an ordered sequence of frames that contain special information and a whole sequence of the video contains main information. So a video classification model with a Convolutional Neural Network – Recurrent Neural Network (CNN-RNN) architecture will be trained on the existing dataset and deployed on the mobile application. This machine learning model will be used to translate the sign language of hearing impaired and inarticulate people into different words and sentences which belong to 29 different classes. This report is structured on different chapters which will describe the complete methodology and design process of the project including a literature review.

Chapter 2

2 Literature Review

Sign language recognition is an important field of research and a lot of related work has been done in this area for different fields. Sign language recognition was done by vision-based approaches, data glove-based approaches, machine learning methods like Convolutional Neural Network, Artificial Neural Network, Fuzzy Logic, PCA, LDA, Genetic Algorithm and SVM, etc. These techniques utilized different approaches consisting of Hand segmentation, Facial expression, Feature extraction, or Gesture recognition. Many researchers have utilized these techniques for sign language recognition. P. D. Rosero-Montalvo proposed a method of sign language recognition based on intelligent gloves using machine learning techniques in which he used hardware-based gloves [1]. The accuracy of the system was 85% and it utilized hardware-based flex sensors to record the sign gestures which resulted in extra cost and the user have to carry additional equipment with him [1]. Mahesh Kumar NB designed a web application-based system to convert sign language into text using Linear Discriminant Analysis (LDA) approach [2]. In this system, the researcher used images of hand sign and extracted features from them to compare with the existing database to classify the sign and generate respective text. Only hand gestures were classified in this system and signs related to other body motions were ignored [2]. M. Mirzaei proposed a vision-based method using Augmented Reality (AR) and Automatic Speech Recognition (ASR) technologies that combine audio, video and facial expression of the signer to translate their sign language [3]. In this system, facial expressions of the subject were recorded and the use of AR technology helped the narrator and signer communicate with ease. But the system was expensive in terms of hardware resources and computational cost [3]. A maker-free, visual Indian Sign Language (ISL) recognition system was developed by Kanchan Dabre in which a machine learning model for sign language interpretation based on webcam images was used to extract features of hand images and classified with Haar Cascade Classifier [4]. This system also used hand gesture images only and not the facial expression or full-body motions. Also, the web application based system made it hard and non-versatile to use in daily life activities [4]. Ankit Ojha presented Convolutional Neural Network (CNN) architecture to recognize the American Sign Language (ASL) for numbers and alphabet [5]. The proposed method used OpenCV for real-time hand pose detection and augmentation and fed these images to the CNN model and trained it to recognize the gesture and convert them

to textual content and speech [5]. This system was developed to classify the American sign alphabets only and therefore the utility of the system was limited [5]. A large-scale video classification model based on a spatial-temporal network and a connected multiresolution architecture was proposed by Andrej Karpathy in which video-based approach was used to classify 1M YouTube videos [6]. Specifically, this proposed architecture was used to recognize human activities and was expensive in terms of computation and performance [6].

In this project, we will also use vision-based approaches like image processing and machine learning to develop a sign language recognition system with easy to use mobile application interface. We will be using video classification to extract frames of video and process them to extract meaningful features. Unlike image classification, videos cannot be easily processed due to their temporal and time-series property. Videos have a fixed-sized architecture that has both spatial and temporal information. Specifically, we will be using a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) architecture for spatial features and temporal features respectively. This architecture is consisting of CNN, GRU and LSTM layers which is a hybrid architecture also known as CNN-RNN. We will also experiment other large-scale video classification models like ConvLSTM which are also hybrid architecture of the same kind. At the end, we will be comparing the results of our developed system with other state of the art models to evaluate the performance and efficiency.

Chapter 3

3 Work Done So Far

3.1 Design Process

After the literature review, we moved towards the implementation of the proposed project. We need to process the video dataset through video segmentation so that we can extract the frames from these videos. After frame extraction, we need to extract the meaningful features from these extracted frames to perform classification. Since video is a series of individual images and therefore we may treat video classification as image classification with a total of N times, where N is the total number of frames in a video. The step-by-step process of this project is explained below.

3.1.1 Data Collection

The first step is Data Collection. We defined the area of interest consisting of categories that we want to address in our project and train the classification model on it. These include words and sentences that are normally used in daily life activities and emergencies. The list of the words and sentence are given below:

Table 1: Dataset recording list

Sr. No	Sentences	Sr. No	Sentences
1	Hi/Hello	**16**	Where are you going?
2	Allah Hafiz/Bye	**17**	I don't understand.
3	Yes/Alright	**18**	What is the time?
4	No	**19**	I need your help
5	Excuse me/I am sorry	**20**	Can you help me?
6	Thank you	**21**	I am looking for this
7	Who are you?	**22**	I cannot speak
8	I am a student	**23**	Clock
9	He is my brother	**24**	Door
10	What is your name?	**25**	Engineer
11	How are you?	**26**	Mechanic
12	I am good	**27**	Emergency
13	Nice to meet you	**28**	Accident
14	Leave me alone	**29**	Call the ambulance
15	I want to go to school/outside	**-**	-

For data collection, the local government-owned school named *Government School for Deaf and Dumb Mianwali* was approached to allow the video recording of deaf-mute students for project purposes. We made the setup and kept normal background and lighting conditions to record videos. The subjects were asked to perform signs and gestures for the tasks listed in table 1. We recorded video samples for each sign and processed them for better use. The picture of the data collection set is shown below in figure 1:

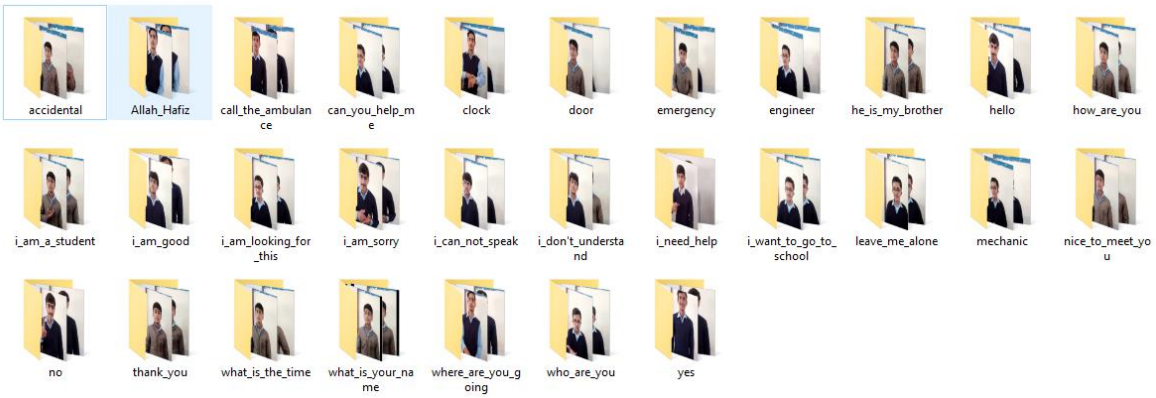


Figure 1: Folders having video sample dataset

3.1.1.1 *Meta Data*

Table 2: Metadata of recorded dataset

No. of Subjects	**10**
No. of Categories/Classes	29
No. of Samples per Category/Class	15-20
Total no. of Samples	**500**

3.2 Mathematical Model

We have used CNN-RNN architecture and ConvLSTM model as explained earlier. These are deep learning algorithms that are very powerful and mostly used in solving computer vision and image or video classification problems. These models are sequential and help to execute functional mathematical operations on dataset and also maintain the long-range sequences in time-series input data. We have utilized these models for video classification as well as maintaining the sequence and temporal information.

3.2.1 Convolutional Neural Network (CNN) Detector

CNN is a powerful class of neural networks that are inspired by the actual perception that takes place in our brain. This is highly useful in solving computer vision problems. It uses

filter/kernel to scan the features of an image and uses convolutional computations to set appropriate weights to detect and classify that specific features. The Keras application offers CNN's pre-trained models like InceptionV3, ResNet50, VGG-16 etc. to extract the features for the input dataset. These models are trained on large amount of dataset and provide transfer learning ability in our code. The performance of these models is high as compared to the model created from scratch. So we have utilized its ore-trained models and trained extracted features in RNN algorithm.

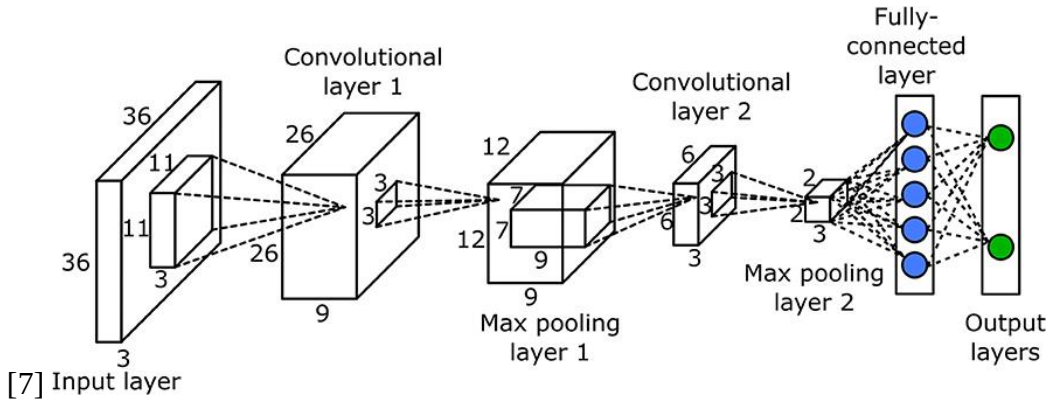


Figure 2: Convolutional layers of an image/frame deep learning model

3.2.2 Recurrent Neural Network (RNN)

RNN is also a type of Neural Network where the output from the previous step is fed as the input for the current step in a layer, Because of this property, RNN is a model used in Sequence Classification or Video Classification problems in which time-series data is involved. We have used this network for temporal processing in our project.

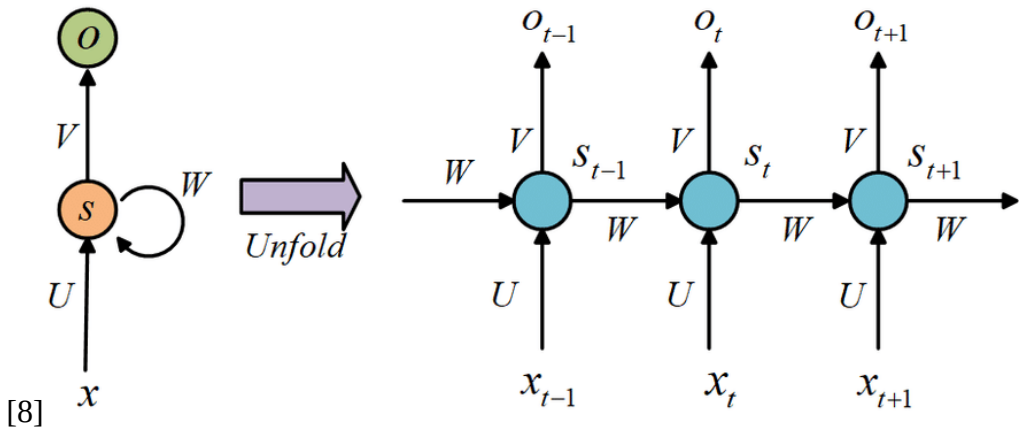


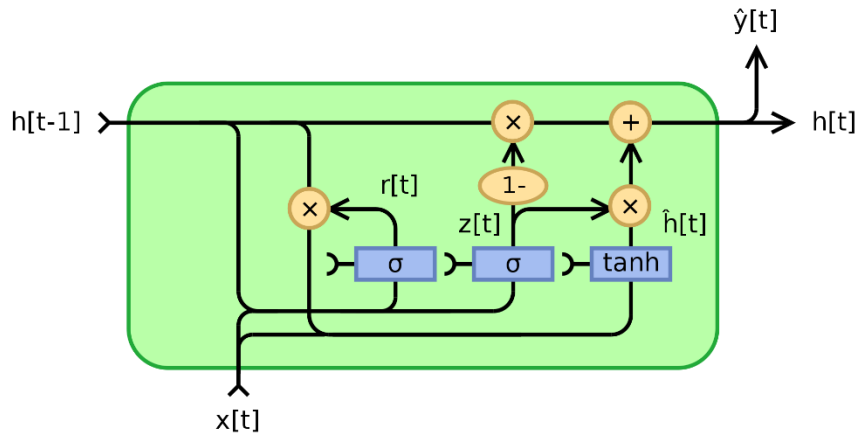
Figure 3: Basic structure of RNN

RNN is very useful in predicting sequence as it accepts current and as well as previous input but the drawback of RNN is that is cannot handle long term sequences. This is called

Vanishing Gradient problem where the sequence of long-range dependencies is forgotten by the model. So, to counter this problem, we have used extended RNN architectures known as Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM). These types of RNN are more powerful in handling long term dependencies and help in handling time-series data for a long sequence of time. A detailed explanation of these architectures are given below:

3.2.2.1 Gated Recurrent Unit (GRU)

This is a very strong model to maintain and remember the sequence in long term decencies in the model. The GRU has different types of gates that enable it to learn the long term sequence. It has a Reset gate and Update gate in its architecture. The model accepts previous input, that is h(t-1) and current input x(t) just like RNN, but it has a special cell called memory cell. This memory cell is updated at every time step in the model. All the inputs combine and give output and as well as generate the new value of the memory cell h(t).

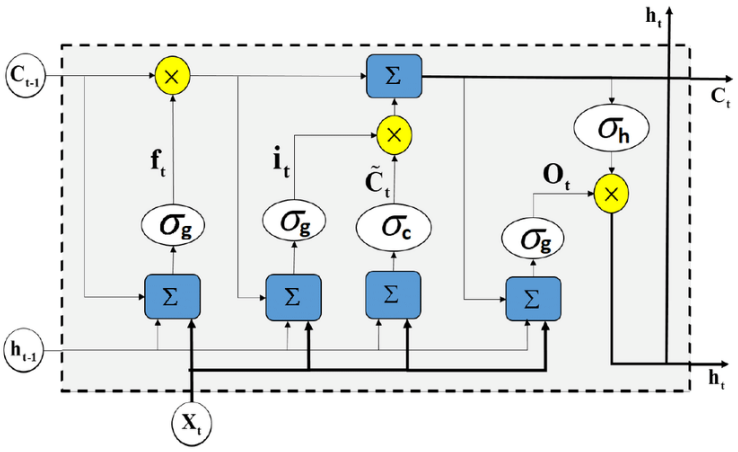


[9]

Figure 4: GRU structural building block

3.2.2.2 Long Short Term Memory (LSTM)

LSTM is another powerful type of RNN and it modulates the flow of information just like the GRU. But the major difference is that it has a Forget Gate. This gate decides which part of information to forget and which part of information to retain and store for tnext time step. It helps to store the most relevant information and forget the irrelevant information. Then the output is passed to sigmoid activation and update gate. The memory cell c(t) updates itself at every time step and act as the previous input for the next time step. This architecture makes LSTM more complex in computations but most powerful at the same time, Whereas, the GRU's simple architecture makes it more efficient and easy to implement.

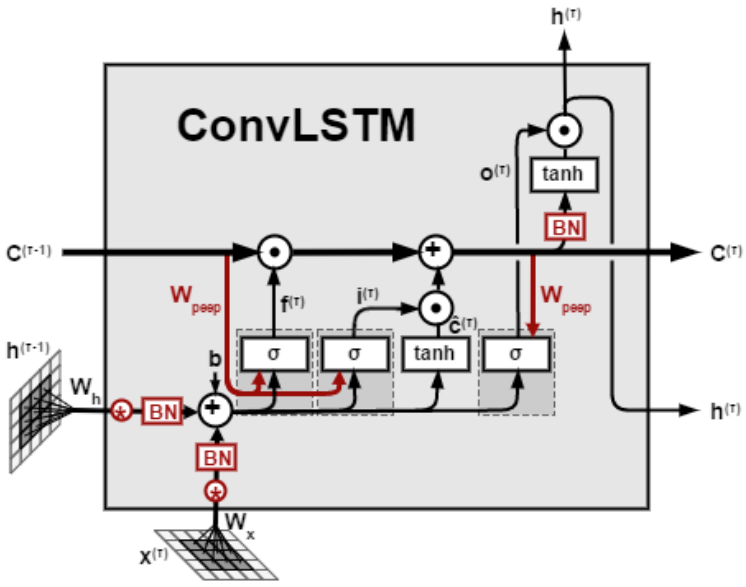


[10]

Figure 5: LSTM structural building block

3.2.2.3 Convolutional Long Short Term Memory (LSTM)

In our project, we are dealing with spatial-temporal predictions. For this kind of dataset, we used ConvLSTM which is a type of recurrent neural network. It has convolutional structure that determines the future state of a certain cell in the input image grid from current input and past input states of its local neighbours. Since, we have a sequence of images in a video dataset so in this case, we have maintained the order and time-series nature of the video using ConvLSTM.



[11]

Figure 6: LSTM cell

3.3 Algorithm

3.3.1 Video Processing and Frames Extraction

The main challenge of building a video classifier is finding out a way to feed the videos to a neural network. There are several methods available for this and we followed the traditional approach. Since video is an ordered sequence of frames so we could just extract the frames in order and put them in a 3D sensor so that they can be fed to the neural network. We had different numbers of frames in each video since the frames differ from video to video. So we saved them at a fixed interval until the maximum frame is reached. This is the problem that is under process and mostly faced on testing videos while performing predictions.

A sample of frames extracted from a video can be seen in figure 4 below. In actual code, we have stored them in a temporary folder to process and extract features from each frame. These features are stored in a feature vector to feed as input for the neural network.

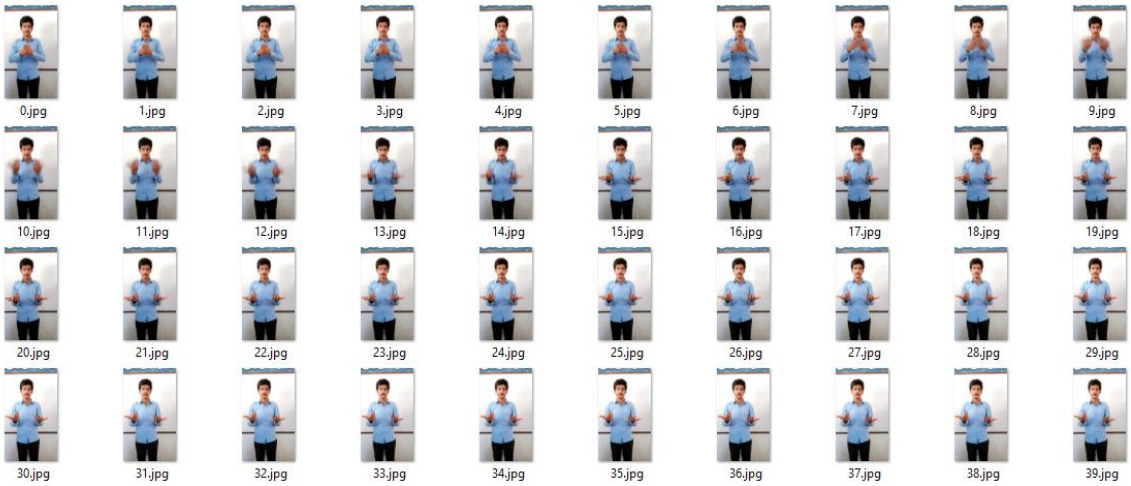


Figure 7: Frames extracted from video sample

We have performed the following steps to process the videos:

1. Capture the frames from the video using the OpenCV library
2. Extract frames from the whole video until the maximum frame is reached.
3. If the video's frame count is less than the maximum frame count, then we need to copy previous frames or use zero padding to match the count length.

3.3.2 Machine Learning Models

For the classification of signs movement and gesture recognition based on captured frames, we needed to train a model on an adequate amount of dataset. For this purpose, we collected our own dataset of sign videos belonging to 29 different categories. We have collected more than 500 sample videos for 29 classes which are not sufficient for the video classification. As we have less amount of sign video dataset so we can even tolerate less accuracy for now.

For starting, we extracted features of video frames for a small number of classes. Several machine learning models can be used for classification problems like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Support Vector Machine (SVM) and Nearest Neighbors (kNN), etc. In this project, we have utilized hybrid architecture which is based on CNN for spatial information and RNN for temporal information. Since we are dealing with time-series data so these are best suited for this problem.

The frames in a video are in an ordered sequence which is related to time series data. We extracted features from the frames and trained our model based on those features. For this purpose, we have utilized the following techniques for features extraction.

3.3.3 Feature Extraction

3.3.3.1 ConvLSTM

ConvLSTM is a hybrid architecture that is used for a spatial-temporal dataset in which we have time-series nature of the data based on images. When working with videos, we face challenge of both classification and sequence. The best approach for images is to pass through Convolutional Layers, in which the filters extract important features. In sequential images, we used ConvLSTM layers, that is recurrent neural network as explained earlier, just like the LSTM, but in this model, the internal matrix multiplication is used which are exchanged with convolutional operations. We used this model and passed our frame vector to its layers. The dimensions were kept 3D by the model instead of a 1D array and we got the output as a sequence of all inputs at each time step. We trained this model and checked performance and accuracy on training and testing datasets. We assigned labels on testing videos and calculated probabilities. The label with the highest probability is then predicated along with other classes after that. The problem with this model is that the model was overfitting on the dataset. The performance was good on training dataset but poor on test/unseen data. We opted for different approaches on this model to reduce overfitting. The results are recorded in the next chapter.

3.3.3.2 ResNet50

Keras Application provides many state of the art models, pre-trained on the ImageNet-1k dataset set and these models are very useful in transfer learning on the dataset. These models are mainly used where we have less amount of data for machine learning models. We have used ResNet50, a pre-trained model for our dataset. The meaningful features are extracted from all frames of all training and testing videos and stored in a feature vector. This vector

is then passed to the sequential model which is based on CNN-RNN architecture. We have used this hybrid architecture consisting of GRU, recurrent neural network layers. We trained this model and checked performance and accuracy on training and testing datasets. We assigned labels on testing videos and calculated probabilities. The label with the highest probability is then predicated along with other classes after that. The results are recorded in the next chapter.

3.3.3.3 InceptionV3

The performance of the ResNet50 was not satisfactory as shown in *chapter 4*. We experimented with another model named InceptionV3 which is also a very popular and useful model for extracting features from the image dataset. We passed the frames dataset to this pre-trained InceptionV3 model and extracted features from all frames of all training and testing videos and stored them in a feature vector. This vector is then passed to the sequential model which is based on CNN-RNN architecture. We trained this model and checked performance and accuracy on training and testing datasets. The performance of this model was satisfactory at this level where we do not have an adequate amount of data. We assigned labels on testing videos and calculated probabilities. The label with the highest probability is then predicated along with other classes after that. The results are recorded in the next chapter.

Chapter 4

4 Results

4.1 Machine Learning Results

The highest probability of the test video sample is shown in the figures below with corresponding video displayed. There are also the probabilities of test video with other classes. Note that the videos are actually frames that we have just extracted and predicted model on these frames. We have utilized these frames and converted them into corresponding video. This is also an indication of validity that our frames are extracted in an ordered sequence.

Now, below are the results of all techniques that we have utilized in our project for sign language prediction.

4.1.1 ConvLSTM

4.1.1.1 Model Accuracy

```
print("Training accuracy: ", history.history['accuracy'][-1]*100)
print("Training Loss: ", history.history['loss'][-1])
print("-----------------------------------------")
print("Validation accuracy: ", history.history['val_accuracy'][-1]*100)
print("Validation Loss: ", history.history['val_loss'][-1])
print("-----------------------------------------\n")

accuracy = convlstm_model.evaluate(features_test, labels_test, verbose=0);
print(f"Test Accuracy: {round(accuracy[1] * 100, 2)}%")
print(f"Test Loss: {round(accuracy[0], 2)}%")
```

```
Training accuracy:  96.55172228813171
Training Loss:  0.3731014132499695
-----------------------------------------
Validation accuracy:  80.7692289352417
Validation Loss:  0.9064874053001404
-----------------------------------------

Test Accuracy: 75.0%
Test Loss: 1.02%
```

Figure 8: Accuracy and Loss of Training, Validation and Testing Dataset

The accuracy of the algorithm is also shown in the following table 3:

Table 3: accuracy and loss of model

Sr#	Dataset	Accuracy (%)	Loss
1	Training	96.5	0.3
2	Validation	80.7	0.9
3	Testing	75.0	1.0

4.1.1.2 Plots of Accuracy and Loss

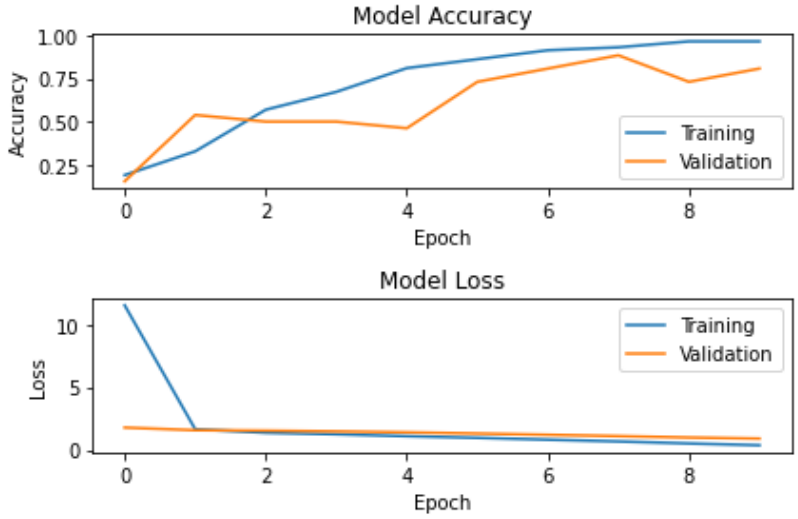


Figure 9: Accuracy and Loss Graph of Training and Validation Dataset

4.1.1.3 Test Run-1

```
test_video = '/content/videos_sample_6/i_need_your_help/i_need_your_help-11.mp4'
print(f"Test video path: {test_video}")
test_frames = sequence_prediction(test_video)

to_gif(test_frames[:SEQUENCE_LENGTH])
```

```
Test video path: /content/videos_sample_6/i_need_your_help/i_need_your_help-11.mp4
i_need_your_help
```

Figure 10: Test video predictions for label "`i_need_your_help`"

4.1.1.4 Test Run-2

```
test_video = '/content/videos_sample_6/call_the_ambulance/call_the_ambulance-2.mp4'
print(f"Test video path: {test_video}")
test_frames = sequence_prediction(test_video)
to_gif(test_frames[:SEQUENCE_LENGTH])
```

```
Test video path: /content/videos_sample_6/call_the_ambulance/call_the_ambulance-2.mp4
call_the_ambulance
```

Figure 11: Test video predictions for label "call_the_ambulance"

4.1.2 ResNet50

4.1.2.1 Model Accuracy

```
print("Training accuracy: ", history.history['accuracy'][-1]*100)
print("Training Loss: ", history.history['loss'][-1])
print("----------------------------------------")
print("Validation accuracy: ", history.history['val_accuracy'][-1]*1
print("Validation Loss: ", history.history['val_loss'][-1])
print("----------------------------------------\n")
_, accuracy = seq_model.evaluate([test_data[0], test_data[1]], test_
print(f"Test accuracy: {round(accuracy * 100, 2)}%")
```

```
Training accuracy:  80.7692289352417
Training Loss:  0.6387647390365601
----------------------------------------
Validation accuracy:  73.9130437374115
Validation Loss:  1.2899788618087769
----------------------------------------

1/1 [==============================] - 0s 226ms/step - loss: 0.9584
Test accuracy: 57.89%
```

Figure 12: Accuracy and Loss of Training, Validation and Testing Dataset

The accuracy of the algorithm is shown in the following table 3:

Table 4: accuracy and loss of model

Sr#	Dataset	Accuracy (%)	Loss
1	Training	80.7	0.6
2	Validation	73.9	1.2
3	Testing	57.8	0.9

4.1.2.2 Plots of Accuracy and Loss

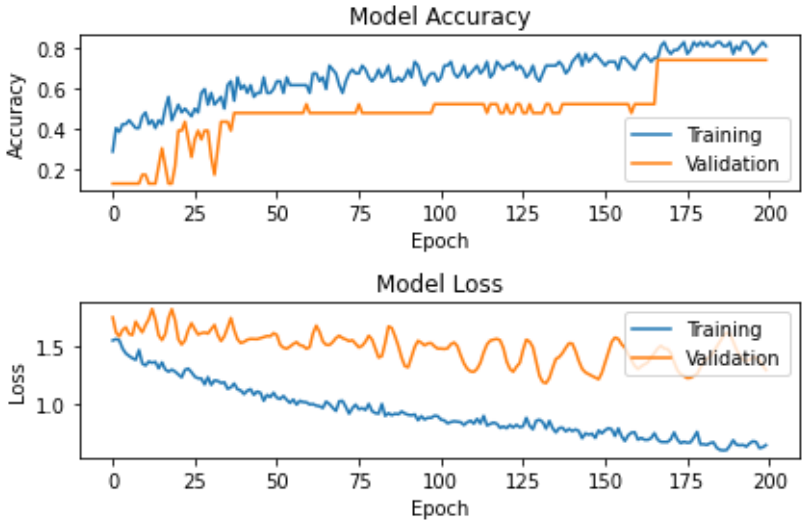


Figure 13: Accuracy and Loss Graph of Training and Validation Dataset

4.1.2.3 Test Run-1

```
test_video = '/content/videos_sample_3/i_can_not_speak/i_can_not_speak-16.mp4'
print(f"Test video path: {test_video}")
test_frames = sequence_prediction(test_video)
to_gif(test_frames[:MAX_SEQ_LENGTH])
```

```
Test video path: /content/videos_sample_3/i_can_not_speak/i_can_not_speak-16.mp4
  i_can_not_speak :0: 69.88%
  i_am_a_student :4: 14.86%
  what_is_the_time :3: 10.43%
  thank_you :1:  3.50%
  call_the_ambulance :2:  1.33%
```

Figure 14: Test video predictions for label "`i_can_not_speak`"

4.1.2.4 Test Run-2

```
test_video = '/content/videos_sample_3/what_is_the_time/what_is_the_time-9.mp4'
print(f"Test video path: {test_video}")
test_frames = sequence_prediction(test_video)
to_gif(test_frames[:MAX_SEQ_LENGTH])
```

```
Test video path: /content/videos_sample_3/what_is_the_time/what_is_the_time-9.mp4
  what_is_the_time :3: 36.46%
  i_can_not_speak :0: 25.11%
  i_am_a_student :4: 16.48%
  thank_you :1: 12.23%
  call_the_ambulance :2:  9.73%
```

Figure 15: Test video predictions for label "what_is_the_time"

4.1.3 InceptionV3

4.1.3.1 Model Accuracy

```
print("Training accuracy: ", history.history['accuracy'][-1]*100)
print("Training Loss: ", history.history['loss'][-1])
print("----------------------------------------")
print("Validation accuracy: ", history.history['val_accuracy'][-1]*10
print("Validation Loss: ", history.history['val_loss'][-1])
print("----------------------------------------\n")
_, accuracy = seq_model.evaluate([test_data[0], test_data[1]], test_
print(f"Test accuracy: {round(accuracy * 100, 2)}%")
```

```
Training accuracy:  84.61538553237915
Training Loss:  0.5661555528640747
----------------------------------------
Validation accuracy:  86.95651888847351
Validation Loss:  0.5741025805473328
----------------------------------------

1/1 [==============================] - 0s 72ms/step - loss: 1.3483 -
Test accuracy: 57.89%
```

Figure 16: Accuracy and Loss of Training, Validation and Testing Dataset

The accuracy of the algorithm is shown in the following table 3:

Table 5: accuracy of Model

Sr#	Dataset	Accuracy (%)	Loss
1	Training	84.6	0.5
2	Validation	86.9	0.5
3	Testing	57.8	1.3

4.1.3.2 Plots of Accuracy and Loss

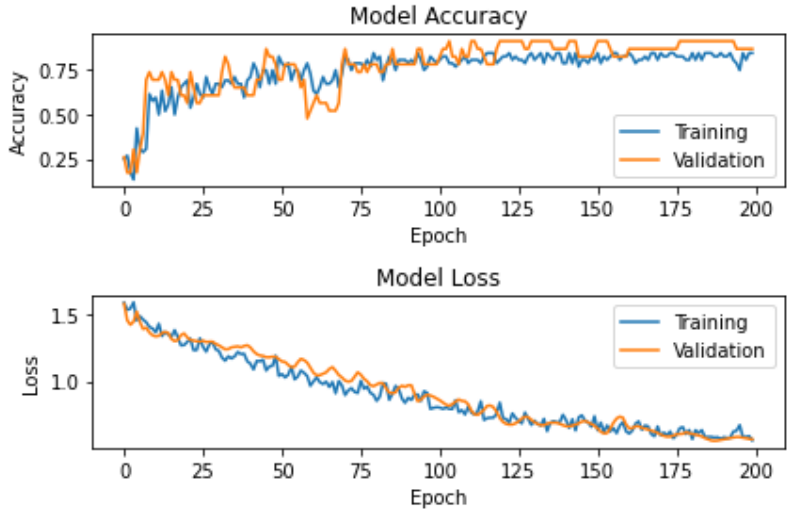


Figure 17: Accuracy and Loss Graph of Training and Validation Dataset

4.1.3.3 Test Run-1

```
Test video path: /content/videos_sample_3/i_am_a_student/
  i_am_a_student :4: 95.60%
  what_is_the_time :3:  1.74%
  i_can_not_speak :0:  1.51%
  thank_you :1:  0.99%
  call_the_ambulance :2:  0.16%
```

Figure 18: Test video predictions for label "`i_am_a_student`"

4.1.3.4 Test Run-2

```
Test video path: /content/videos_sample_3/i_can_not_speak
  i_can_not_speak :0: 89.88%
  call_the_ambulance :2:  5.48%
  i_am_a_student :4:  3.17%
  thank_you :1:  1.10%
  what_is_the_time :3:  0.37%
```

Figure 19: Test video predictions for label "`i_can_not_speak`"

Chapter 5

5 Conclusion

The training and testing of the algorithm are under process and we are trying to increase the dataset for better accuracy of the model so that it has minimum loss when deployed on the mobile application. So far, we have invested our time in data collection for the project which was time-consuming and then building the model for the available dataset. We have performed video processing separately and then incorporated it with the model to pass videos directly in the code. We have trained the model for less number of classes with more datasets. This means if we increase the dataset per class, we would get much better accuracy of the model and predictions will be done correctly. Finally, we'll be able to embed this whole system with a mobile application that can be easily used by the hearing impaired and inarticulate people to convey their message to the ones who don't have any disability. Moreover, people with no hearing and speaking disabilities will also be able to use this mobile application to understand the sign language of disabled people.

Chapter 6

6 Way Forward

The project is almost at the position from where a little more work will be required to get the best possible accuracy of the machine learning model. The first major task to do is to increase the dataset and record the dataset for more categories and increase the volume of video samples per category. By having this huge dataset, we can train the CNN-RNN model which will increase the accuracy of the system. For this purpose, we will need more processing power and may have to buy online cloud processing services like AWS and Amazon cloud. After getting satisfactory results, we will shift to designing and building the mobile application interface and work on its architecture to deploy the model successfully so that people can use this application with ease and portability.

When we'll be having a good number of datasets, the next step will be to pre-process the video samples so that we can get a cleaned dataset that helps the model t perform even better to give us the correct results. After pre-processing the dataset, the next step would be to implement the pre-trained CNN-RNN model. The model's hyper parameters can be changed to get good accuracy on training, validation, and testing. As a result, we would be able to correctly predict the unseen data from the trained model. Hence by this, we would generalize our model for more classes to make it generalized.

After training the CNN-RNN model on more classes a generalized system will be developed. Now the next step would be to embed this whole system with a mobile application to provide the people with hearing and speaking disabilities and the ones with no disabilities, an easy user interface to convey and understand the sign language respectively. Mobile application development will be started after making our model a generalized one. An easy user interface will be used for the mobile application so that the user doesn't feel any difficulty in using the mobile application. Hence by the end of this phase, when we'll embed the generalized trained model with an easy user interface mobile application, we'll be having a commercialized product that can be easily accessible to people.

7 References

[1] P. D. Rosero-Montalvo et al., "Sign Language Recognition Based on Intelligent Glove using Machine Learning Techniques," in *IEEE Third Ecuador Technical Chapters Meeting (ETCM)*, 2018, pp. 1-5, doi: 10.1109/ETCM.2018.8580268.

[2] Mahesh Kumar, "Conversion of Sign Language into Text" in *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 13, Number 9 (2018) pp. 7154-7161

[3] Mirzaei, M.R., Ghorshi, S. & Mortazavi, M. "Audio-visual speech recognition techniques in augmented reality environments." *Vis Comput* 30, 245–257 (2014). doi: 10.1007/s00371-013-0841-1.

[4] K. Dabre and S. Dholay, "Machine learning model for sign` language interpretation using webcam images," in *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, 2014, pp. 317-321, doi: 10.1109/CSCITA.2014.6839279.

[5] A. P. S. M. A. T. D. D. P. Ankit Ojha, " Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network," in *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCAIT*, vol. 8, no. 15, 2020.

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1725-1732, doi: 10.1109/CVPR.2014.223.

[7] Convolutional Neural Network [Online]. Available: https://brilliant.org/wiki/convolutional-neural-network/

[8] Basic RNN Structure [Online]. Available: https://www.researchgate.net/figure/Basic-RNN-structure_fig1_334464982

[9] GRU: https://stackoverflow.com/questions/55261557/understanding-gru-architecture-keras

[10] LSTN: https://www.researchgate.net/figure/A-typical-architecture-of-a-long-short-term-memory-LSTM-cell-An-LSTM-block-typically_fig1_344213150