
ABDULLAH ZAFAR
**01-134162-052**
MUHAMMAD ARSLAN ISHAQ
**01-134162-025**

# Silent Dictation using Lip Reading

**Bachelor of Science in Computer Science**

Supervisor: Ma'am Maryam Bibi

Department of Computer Science
Bahria University, Islamabad

April 2020

# Abstract

Visual speech recognition is a relatively new research field where speech is recognized through the movement pattern of lips. This technique is applied in scenarios where audio is not available or is unclear, or the user cannot or prefers not to use his voice. The aim of this project is to recognize the word spoken by a person in a video without the aid of audio. The developed application first reads a video and converts it to frames. Then for each frame it detects face, then face landmarks and then extract lips region. These lips region frames are pre-processed before they are provided to the deep learning model for classification the result of which is displayed on the screen.

# Acknowledgments

In the name of Allah the Most Gracious and the Most Merciful. It is all due to his blessings that Alhamdulillah we have completed this project. We would like to express our deepest thanks and gratitude towards Ma'am Maryam Bibi who gave us the chance to work on this project, her supervision and her guidance proved very successful in making this project successful. She provided us with the reference material and many other project related materials that not only helped us making the project without any considerable trouble but also completing it on time. We would also like to acknowledge and thank Ma'am Shehla Saif for her continuous guidance and her extensive knowledge of the subject that helped us a lot in the project and was crucial in its completion. We would also like to acknowledge our friends Danyial Tariq, Bilal Hussain, Usama Qazi, Rao Muhammad Ammar Khalid, Umer Azam, Muhammad Arsalan Aftab, Usama Habib and Muhammad Saleh who took part in the development of our custom made dataset. We offer our deepest gratitude to them all. May ALLAH bless them all (Ameen).

ABDULLAH ZAFAR, MUHAMMAD ARSLAN ISHAQ
Bahria University Islamabad, Pakistan

April 2020

# Contents

# List of Figures

# List of Tables

# Acronyms and Abbreviations

| | |
|---|---|
| CNN | Convolutional Neural Network |
| LSTM | Long Short-Term Memory |
| HOG | Histogram of Oriented Gradients |
| SVM | Support Vector Machine |

# Chapter 1

# Introduction

## 1.1 Project Background/Overview

With the advancements in technology, efforts are being made to provide people with services that are improved in quality and reliability. One example of a service is speech recognition through traditional method which involved the use of a microphone. It allows the users to dictate text or give commands verbally to a device. However, it is believed that humans, while listening, do not solely rely on hearing but also on the visuals to correctly grasp whatever is being said. This happens only in cases where the face of the speaker is in the sight of the listener. The same concept of recognizing speech through visuals using technology is called visual speech recognition or lip reading. When it comes to traditional voice-based speech recognition, remarkable improvements have been made in the past decade. Visual speech recognition is, however, a relatively new subject.

Although recognition of words from audio have improved and achieved great accuracy, it still struggles in noisy environments [1, 6, 4]. Some films do not have audio called silent films. Sometimes in a video, the audio is removed due to copyright violations or the audio gets corrupt or the noise in the audio makes the saying of a speaker inaudible. To resolve issues like these, a new technique was required. This new technique was to recognize speech through the visual movement of lips called lip reading [3]. Although, it is most commonly used by deaf and hard of hearing people, it is also used unconsciously by people with normal hearing provided the speaker is in front of them [7].

Visual speech recognition in combination with verbal speech recognition can improve speech recognition in noisy environments. Lip reading can be used for profanity detection, play prediction in sporting events [2] and provide dictation to device facility using lip reading in situations where verbal dictation is not convenient or is unwanted such as in an ongoing event in an auditorium. Deaf or hard of hearing people can use lip reading to figure out what people are saying in videos where captions are not available and sign language was not used such as news on television which often does not provide captions or sign language.

Recognition of speech is an old problem but people with damaged voice box still cannot use the facility. Deaf people mostly use sign language. Most cannot speak correctly due to missing speech information. In some situations, verbal dictation to a device is undesirable. However, they can replicate the lip movements for particular words. In order to address these problems, the software application will help solve these problems to some extent by visual speech recognition of a limited set of vocabulary.

It is important to note that whenever a person talks, the lips of the person always move in a similar pattern for a particular word. This pattern can be used to predict the word uttered by a person. An interesting problem arises when phonemes are discussed with regards to lip reading. The lip movement pattern for phonemes are quite similar and therefore it can be a difficult and challenging conundrum to accurately predict the word. For example, the lip movement pattern is very similar for the words cat and hat.

Machines tend to perform better and faster than humans. It is riveting to see how this hypothesis fares against the lip-reading conundrum. Human beings are not very accurate when it comes to lip reading especially without any audio input. However, computers too may find lip reading a very perplexing and difficult task as the movement pattern of lips for many words are similar. In the fields of computer, lip reading has become a trendy research topic recently due to its potential applications and their potential to serendipitously be among the marvels of technology.

The project is to develop an application which uses lip reading to predict the word being said in the short audio-less video and generate text therefore enabling the user to dictate one-word text to the device using lip reading only. Due to the complicated nature of the task of lip reading, the project is to be restricted to visual speech recognition at word level with limited vocabulary.

## 1.2 Problem Description

In noisy environments, voice-based speech recognition is affected by a loss of accuracy. Secondly, deaf people cannot utilize the facility of dictating to the device as their speech is impaired. Often, we come across situations where speech recognition is required but is affected by noisy audio or sometimes audio is not even available or verbal dictation is undesirable. For such situations, a new technique for speech recognition is required which is through lip-reading.

## 1.3 Project Objectives

The objective of the project is to develop an application which can identify the word being spoken by a user given only the video input. The application shall be designed for environments where clear or noise free audio is unavailable or verbal dictation is undesirable, and the sole reliance is on video inputs.

## 1.4 Project Scope

This application aims to achieve speech recognition through lip-reading and enable word level dictation to the device in any environment without the aid of audio. The video must consist of a human face speaking a word from limited vocabulary. Our project should be able to recognize the word through lip-reading and after successful recognition, it will output that word as a result on the screen. The application will not be able to recognize more than one word per video. The application will accept short fixed length videos only.

# Chapter 2

# Literature Review

Speech recognition has been around for a while, embedded in innumerable application software to provide conveniences to users. It has been gradually improving over time. Initially, speech recognition systems used machine learning models but later they were made using deep learning models. This significantly improved accuracy. The accuracy rates of speech recognition, as claimed by tech giants Google and Microsoft, are close to or have achieved the 95% mark.

Most of the work performed in the recent years on lip reading, to make it more palatable, is based on machine learning [2]. More recently, various deep learning methods have been applied to solve this problem with better results. Some methods have been discussed below:

Yuanyao et al. [1] proposed a model for lip reading based on deep Convolutional Neural Network (CNN) [8] and attention-based Long Short-Term Memory (LSTM) [9] as shown in Figure 2.1.

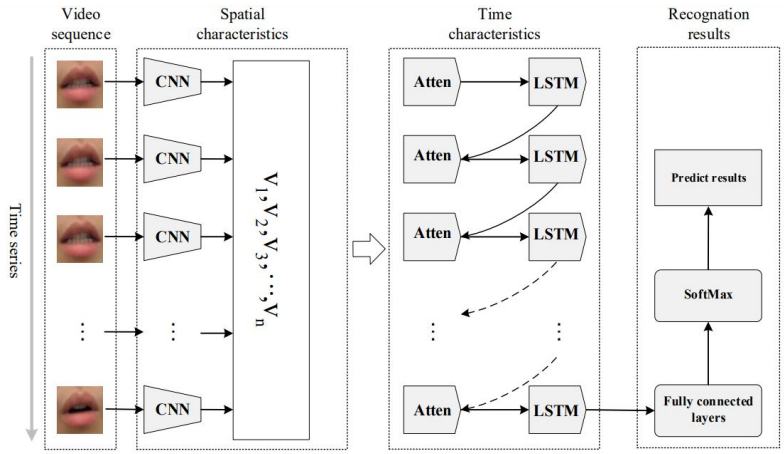


Figure 2.1: Architecture of Convolutional Neural Network (CNN) with attention-based Long Short-Term Memory (LSTM) [1].

Videos are sequential images. The videos were recorded at 25 frames per second and each

utterance may have different length. The authors first preprocessed the sequential images of lips.The time of utterance was divided into 10 equal portions. A random frame from each portion was selected as keyframe. This made each utterance to be of equal length. Face landmarks were then detected from each image and mouth areas were segmented. These images were resized to 224 x 224 pixels. These preprocessing steps are displayed in Figure 2.2.

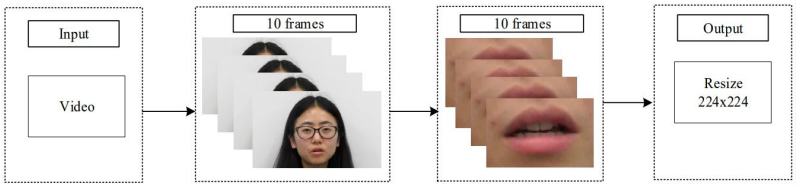


Figure 2.2: Preprocessing steps performed by Yuanyao et al. [1].

The authors then used the model based on VGG19 [10] without the last two fully connected layers. They continued training the model based on pre-training parameters of ImageNet. This CNN [8] part was followed by LSTM [9] with attention mechanism, two fully connected layers and a SoftMax layer which assigns probability to each result such that the sum of probabilities for all results is one. The visualization of feature extraction using the CNN [8] part is shown in Figure 2.3 along with the layers that follow.

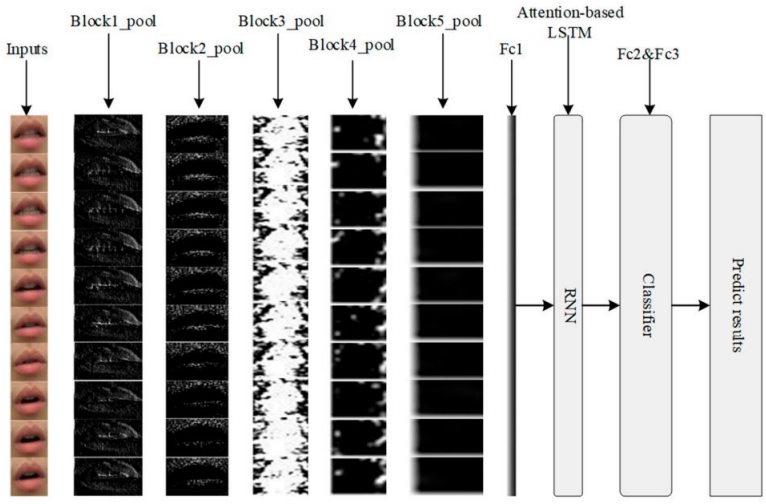


Figure 2.3: Visualization of feature extraction using VGG19 [1].

The dataset used for the experiment was created by the authors themselves. It consisted of 3 male and 3 female speakers. Each speaker uttered the numbers zero to nine hundred times each. 80% data was used for training whereas the remaining 20% was used for testing.

The accuracy of this model was 88.2% whereas a simple CNN [8]-LSTM [9] network gave 84.9% accuracy on the same dataset. Figure 2.4 shows comparison between accuracy achieved on each word by proposed attention-based CNN [8]-LSTM [9] and simple CNN [8]-LSTM [9].

Abiel et al. [2] proposed 4 models. The authors used MIRACL-VC1 dataset consisting of 15 speakers including 10 females and 5 males. There are 10 words and 10 phrases uttered 10 times

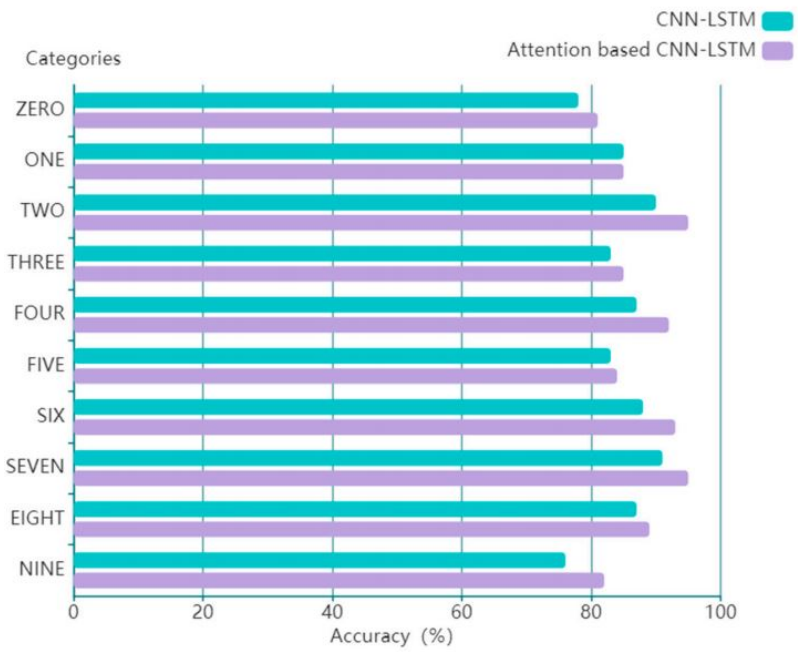


Figure 2.4: Accuracy comparision of attention-based CNN+LSTM and CNN+LSTM on each word [1].

each by each person. Videos had varying lengths. The dataset also consists of depth images of each frame. The authors ignored the phrases and depth images and considered only the videos in which words were spoken.

For preprocessing, the authors first used a pre-trained face detection model to get the face bounding box for each frame in all videos. The face areas were then cropped to remove all the unnecessary background from the frames. These face-only frames were resized to 90 x 90 pixels as the models require uniform input sequences.These steps are shown in Figure 2.5. Data augmentation was performed by horizontal flipping and pixel jittering of each image to increase the size of dataset.

Figure 2.5: Preprocessing of a single frame [2].

The first model proposed by the authors is CNN [8]+LSTM [9] Baseline. It consisted of CNN [8] with depth of 3 and kernel size of 5 x 5 which accepted sequences of images as input. The flattened output of the CNN were fed as a sequence into LSTM [9] after which a fully connected layer was added with 10 units and softmax activation which produced probabilities. The architecture

of this model is shown in Figure 2.6. Adam optimizer was used whereas loss function was cross entropy.

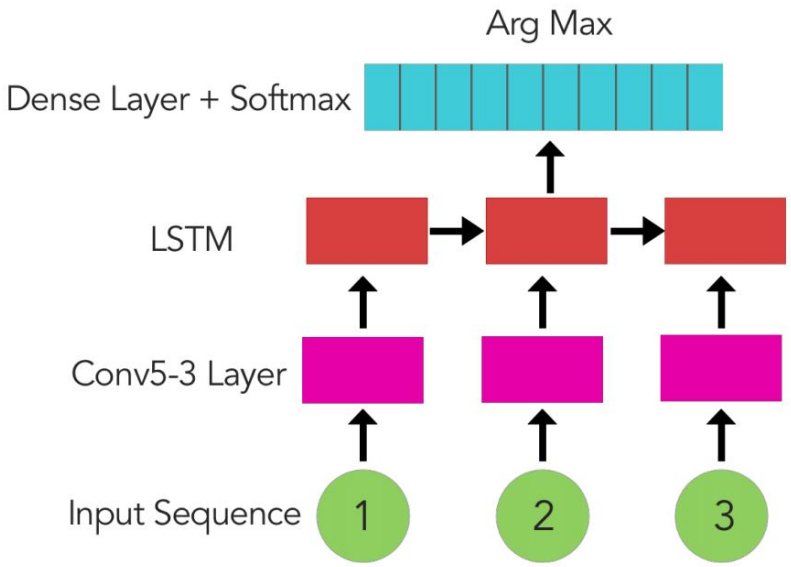


Figure 2.6: CNN + LSTM Baseline model layer architecture diagram [2].

The second model proposed by the authors is deep layered CNN [8]+LSTM [9]. It was an expansion of the first model. 2 additional CNN [8] layers were added. After each CNN layer, dropout with probability 0.2 and batch normalization was added. Max pooling layer with strides of 2 were also added between CNNs [8]. Instead of LSTM [9], bidirectional LSTM [9] was used. Figure 2.7 shows the architecture of this model.

The third model proposed by the authors is based on pre-trained VGG16 [11] on ImageNet. Features were extracted until the last CNN [8] layer and the result were fed into the bidirectional LSTM [9] and then the fully connected layer. VGG model weights were frozen and only weights of LSTM [9] and fully connected layer were being updated.

The fourth model proposed by the authors is Fine-tuned VGG16 [11] + LSTM [9]. They unfroze the last convolutional block of pre-trained VGG16 [11] model. Instead of using Adam optimizer, they used regular SGD optimizer as it is subtler in changing weights.

The tests were conducted for seen and unseen speakers. All models performed rather poorly on unseen speaker with accuracy around 10%. However, for seen speakers the accuracies achieved for the 4 models were relatively much better. The first model, Baseline CNN [8] + LSTM [9], achieved 85% on training, 64% on validation and 39% on test set. The second model performed worse than the first model achieving 52% accuracy on training, 39% on validation and only 25% on test set. Improvement in accuracy was observed when transfer learning was used in the third model. This model achieved 100% accuracy on training, 76% in validation and 55% in test set. Fine-tuning the VGG proved fruitful in the fourth model achieving the highest accuracy among all four models with 100% on training, 79% on validation and 59% on test set.

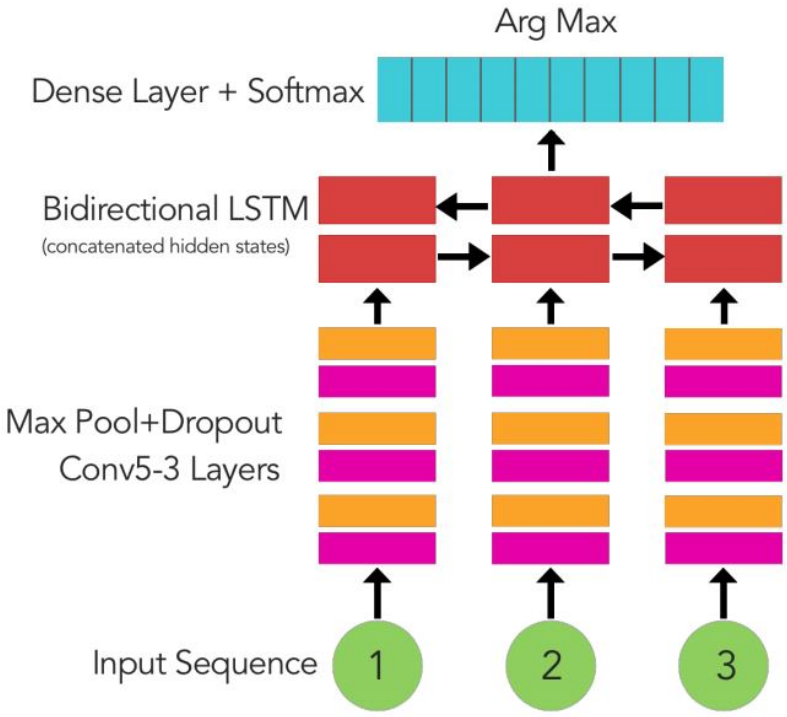


Figure 2.7: Deep Layered CNN + LSTM model layer architecture diagram [2].

Due to lack of availability of large datasets for the purpose of lip reading, Joon et al [3] proposed a system for automatically collecting data. They selected news and debate programs as their source of obtaining data to obtain a variety of speakers speaking different words. Techniques to align subtitles with audio were used to get the time stamp for each word. The authors then performed shot boundary detection, face detection using Histogram of Oriented Gradients (HOG) and tracking as shown in Figure 2.8. Face landmarks were detected in each frame to determine the position of the mouth. The videos were accepted only if they were in sync. One second video of the word was considered regardless of the fact that it may take more or less time to pronounce any word.

Figure 2.8: Left: Face detections; Middle: KLT features and the tracked bounding box (in yellow); Right: Facial landmarks. [3].

The authors used VGG-M model (Figure 2.9) as base and proposed 5 architectures for lip reading namely Early Fusion (EF-25), Multiple Towers (MT-1), Multiple Towers (MT-5), Late Fusion (LF-5), Long Short-Term Memory (LSTM-5). These architectures are shown in Figure 2.9.

The details of these are given in [3] and will not be discussed here. The created dataset was called LRW.

The authors have mentioned top-1 and top-10 prediction accuracies. There were two kinds of input sequences tested by the authors. One was continuous where the one second window maybe have slight portion of neighboring words. The second was isolated where strictly one word would appear in the video using forced alignment output making it possible for word to last less than a second. For continuous and top-1, EF-25 achieved 57% accuracy, MT-1 had 61.1%, MT-5 got 66.8%, LF-5 attained 65.4% and LSTM-5 achieved 66%. For continuous and top-10, EF-25, MT-1, MT-5, LF-5 and LSTM-5 achieved 88.8%, 90.4%, 94.6%, 93.3% and 94.3% accuracies respectively. Isolated performed better than continuous. For top-1 prediction on isolated, EF-25, MT-1, MT-5, LF-5 and LSTM-5 achieved 62.5%, 64.2%, 69%, 68.2% and 71.5% respectively. For top-10 predictions on isolated, EF-25, MT-1, MT-5, LF-5 and LSTM-5 achieved were 92.6%, 94.2%, 95.6% 94.8%, 96.4% respectively. From the results, it can be concluded that the accuracies for top-10 predictions are higher whereas isolated perform better as the neighbor words do not interfere and cause ambiguity in the learning process.

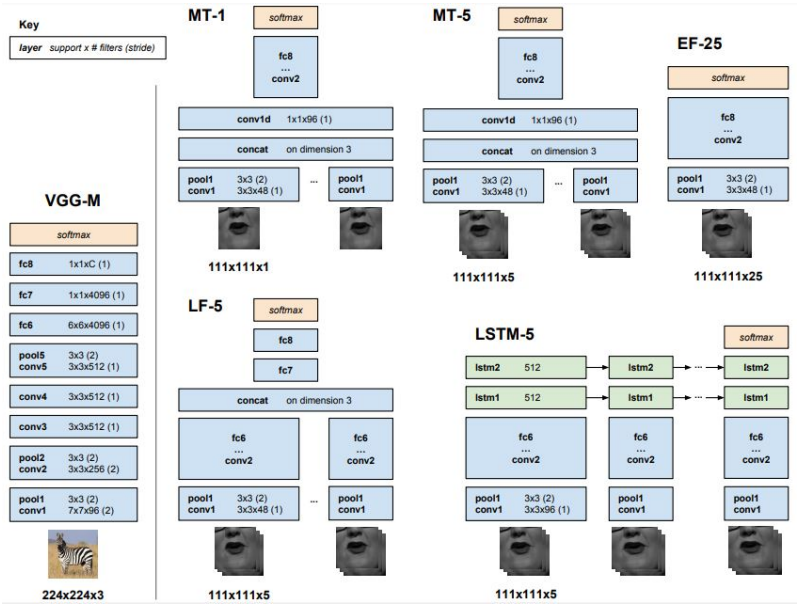


Figure 2.9: VGG-M architecture that is used as a base and five architectures for lip reading [3]

Some work has gone beyond word level and have targeted recognition of entire sentences through lip reading. Joon et al. [4] suggested a new Watch, Listen, Attend and Spell (WLAS) network. This network learns to predict characters in sentences. The image encoder 'Watch', audio encoder 'Listen' and character decoder 'Spell' are the three key components of WLAS network. The image encoder consist of CNN [8], based on VGG-M and ConvNet, and LSTM [9]. The audio encoder simply consist of LSTM encoder. The character decoder is based on LSTM [9] transducer. A dual attention mechanism was added here. The authors proposed WLAS architecture is shown in Figure 2.10.

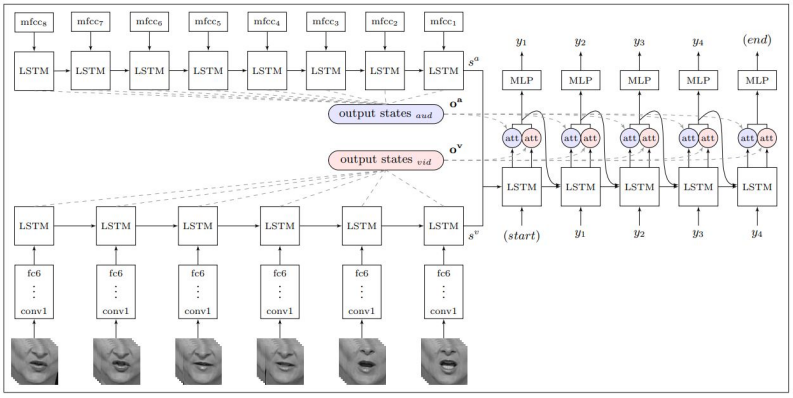


Figure 2.10: Watch, Listen, Attend and Spell architecture [4].

The dataset used was Lip Reading Sentences (LRS) which is an audio-visual dataset. This dataset was extracted in similar way to the already discussed method of Joon et al.[3]. The authors used various training strategies including Curriculum learning, Scheduled sampling, Multi-modal training, Training with noisy audio discussed in detail in [4].

The authors provided with results in form of Character Error Rate (CER) and Word Error Rate (WER). We will only consider results of lip reading without audio. With simple WAS (Watch Attend Spell) CER was 59.9% and WER was 76.5%. WAS + Curriculum learning had a CER of 47.1% and a WER of 61.1%. WAS + Curriculum learning + Scheduled sampling had a CER of 42.4% and a WER of 58.1%. WAS + Curriculum learning + Scheduled sampling + Beam Search had a CER of 39.5% and WER of 50.2%.

Daehyun et al. [12] used OuluVS2 dataset containing videos of 52 speaker. The videos were recorded from five different camera views simultaneously. Each speaker spoke each phrase 3 times. The authors first pre-processed this dataset. All image frames were resized to have same rectangular size of 20 x 20. The contrast was maximized and pixel values were normalized.

The authors experimented 2D-CNN [8] and 3D-CNN [8]. Third experiment consisted of making each image to have 15 channels by adding RGB channels of all 5 views. In the fourth experiment, the authors combined frames captured at same time from different angles into one image. The last experiment was to merge features extracted from 2D-CNN [8] of images taken at same time from different views. The models consisted of two LSTM [9] layers and then a fully connected layer. Adam optimizer was used and loss function was cross entropy.

The authors obtained a test accuracy of 76.1% on 3D CNN [8]. Merge channels technique got 78.9% accuracy. Merge images techniques obtained an accuracy of 80.0% whereas Merge features technique achieved 79.4% accuracy.

Ahmed et al. [5] proposed using RGB-D cameras for lip reading. General overview of the proposed system is given in Figure 2.11. The authors used depth images in addition to regular RGB images. 3-D rigid face tracking method was used to track face position and orientation in order to accurately extract lip region. Appearance and motion descriptors (to modelize lip movement) were combined to modelize temporal deformations of the mouth. The authors chose HOG for

appearance descriptors whereas they tested Internal Motion Histograms (IMH) variant IMHcd and Motion Boundry Histogram (MBH) for motion descriptors. Support Vector Machines (SVMs) were selected for classification purpose. Tested kernels include linear, polynomial and RBF where best performance was given by linear kernel. Although the tests were conducted on OuluVS and CUAVE datasets as well, the main focus of the authors was on MIRACL-VC dataset.

The authors test their models in two different setting. Speaker independent (SI) and speaker dependent (SD). We will first discuss accuracies on words in both settings. The HOG descriptor on colored images lead to an accuracy of 92.8% in SD setting whereas 48.1% on SI setting. HOG descriptor on depth images gave 94.2% and 54.2% on SD and SI settings respectively. Using HOG on both colored and depth images attained 95.4% and 59.8% on SD and SI setting respectively. MBH on colored images gave 86.2% and 45.1% on SD and SI settings respectively. HOG on colored images + HOG on depth images + MBH on colored images gave 95.3% and 63.1% accuracies on SD and SI settings respectively.

On phrases, the HOG descriptor on colored images lead to an accuracy of 95.4% in SD setting whereas 54.4% on SI setting. HOG descriptor on depth images gave 94.2% and 64.7% on SD and SI settings respectively. Using HOG on both colored and depth images attained 96.0% and 66.7% on SD and SI setting respectively. MBH on colored images gave 87.4% and 49.4% on SD and SI settings respectively. HOG on colored images + HOG on depth images + MBH on colored images gave 96.0% and 79.2% accuracies on SD and SI settings respectively.

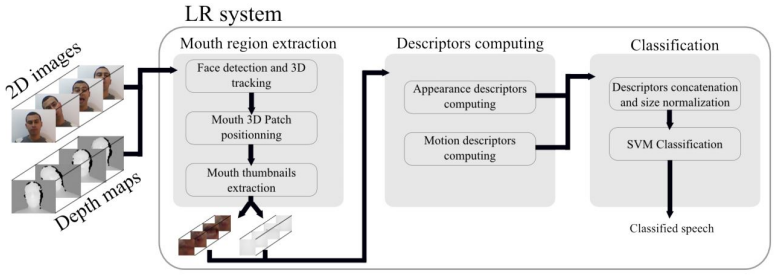


Figure 2.11: Architecture of the system proposed by Ahmed et al. [5].

# Chapter 3

# Requirement Specifications

## 3.1 Existing System

In this section, we will look at the work that has already been done on this field. A lot of the work involved machine learning methodology and did not touch upon the deep learning methods, that are now becoming more common practice because of their exceptional results. Even though research has been bloomed in this particular field, software applications have yet to be made to fully utilize its applications. LipNet, developed by Google's AI division, is the best lip-reading tool for word and sentence detection, but there are no proper applications available to public utilizing it. Lipreader program is also a software made by a software company in UK, seeking to help people with hearing loss, but it is also not available commercially. Due to lack of proper applications available commercially, we cannot evaluate any, to benchmark against our own application.

## 3.2 Proposed System

Our system works on the principals of deep learning methodology to produce results. MIRACL-VC1 dataset will be used containing videos of fifteen people speaking ten words ten times. We pre-process the data by using an existing pre-trained facial recognition model to detect the face and then use a pre-trained face landmarks detection to crop around the subject's lips in every frame of the video and use these frame sequence as input to model. We then explore baseline CNN [8]+ LSTM [9] model, a Deep Layered CNN [8]+ Bi-LSTM [9] model, VGG-16 [11] + Bi-LSTM [9] model, VGG19 + Bi-LSTM [9] model and ResNet50 + Bi-LSTM [9] model. We choose a model that achieves highest accuracy in word recognition and thus, enabling it to identify words from the videos having less than or equal to specified duration. The system will be a mobile application which is able to read video from camera or gallery, process the video and recognize the word spoken in the video through the model. It will be the first application that truly benefits from this field of work.

## 3.3 Requirement Specifications

The functional and non-functional requirements of the application are specified below:

### 3.3.1 Functional Requirements

- System should be able to analyze video and read frames.
- System should be able to identify face through a face detection model.
- System should be able to extract lip region from the face in the original frame.
- System should be able to identify words through a trained model.
- System should be able to generate the identified word as an output.

### 3.3.2 Non-Functional Requirements

- **Reliability:** The system should be reliable in detecting word from the given videos
- **Maintainability:** Developers will be responsible for system maintainability.
- **Availability:** After installation of the application on the smartphone, it will be available for use.
- **Re-usability:** Different modules of the system will able to be reusable in some other systems.

## 3.4 Use Cases

Our application is aimed at general public and people belonging to various other fields that can benefit from our application. Goal of the user would be to use the application to identify word after giving video as an input to the application. The user could use it to understand the word from small videos having no audio at all or from videos having noisy environment. User would need an input in the form of the video which can be acquired from video gallery or camera and then that video would be processed in order to identify word through a classification model that would display the word on screen after successful recognition. General use case diagram is shown in Figure 3.1

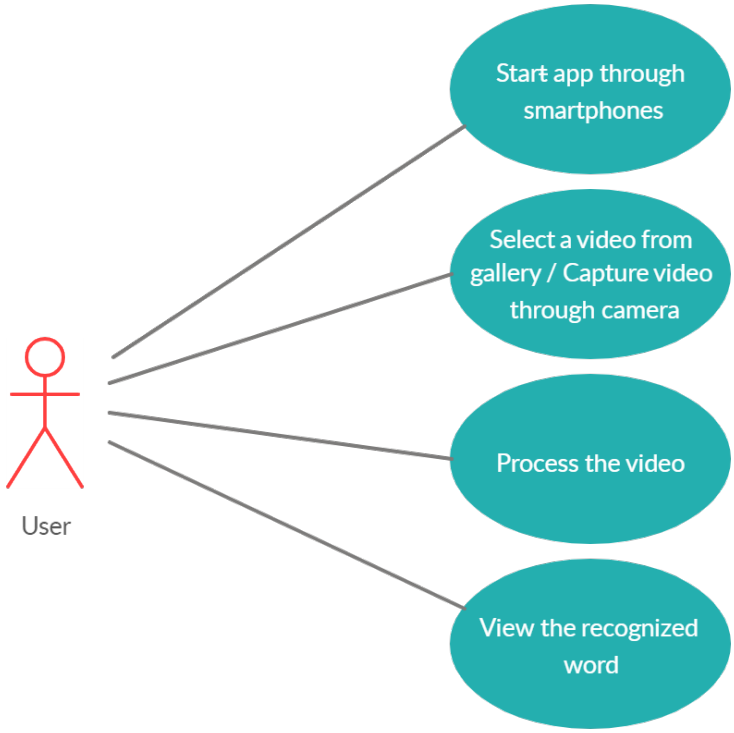


Figure 3.1: Use Case Diagram

### 3.4.1 USE CASE 1: Start App through smartphone

The following Figure 3.2 and Table 3.1 shows how to start the application through your smartphone.


Figure 3.2: Use case 1: Start App through smartphone

| | |
|---|---|
| Title | Start application through smart phone |
| Actor | User |
| Description | This use case represents the interaction between user and the smartphone to start the application |
| Main success factor | Application should start after touching the application icon on screen |
| Pre-condition | Android should be running |
| Post-condition | The application should display a home screen after successful start-up |

Table 3.1: Start App through smart phone

### 3.4.2 USE CASE 2: Select video form the gallery

The following Figure 3.3 and Table 3.2 shows how we can select video from the gallery for further processing.


Figure 3.3: Use Case 2: Select video form the gallery

| Title | Select video from gallery |
|---|---|
| Actor | User |
| Description | This use case specifies the action of choosing a video by the user |
| Main success factor | Application accepts the desired video chosen by the user |
| Pre-condition | Application should be running |
| Post-condition | The application should display the chosen video on the screen implying its acceptance |

Table 3.2: Select video from gallery

### 3.4.3 USE CASE 3: Record video using camera

The following Figure 3.4 and Table 3.3 shows how we can record video through our smartphone camera for processing.


Figure 3.4: Use Case 3: Record video using camera

| Title | Capture video through camera |
|---|---|
| Actor | User |
| Description | This use case specifies the action of capturing video through the camera |
| Main success factor | Application accepts the video recorded by the user |
| Pre-condition | Application should be running |
| Post-condition | The application should display the recorded video on the screen implying its acceptance |

Table 3.3: Record video using camera

### 3.4.4 USE CASE 4: Process the video

The following Figure 3.5 and Table 3.4 shows how we can process the selected video.


Figure 3.5: Use Case 4: Process the video

| Title | Process the video |
|---|---|
| Actor | User |
| Description | This use case specifies the interaction of the user with application for the processing of the video |
| Main success factor | Video should start to process successfully |
| Pre-condition | There should be a video selected by the user |
| Post-condition | The video is successfully processed |

Table 3.4: Process the video

### 3.4.5 USE CASE 5: View the recognized word

The following Figure 3.6 and Table 3.5 shows how we can view the identified word by the application.


Figure 3.6: Use Case 5: View the recognized word

| Title | View the recognized word |
|---|---|
| Actor | User |
| Description | This use case specifies the action of viewing the recognized word by the user |
| Main success factor | User views the recognized word after processing |
| Pre-condition | The video should be processed |
| Post-condition | The word is displayed on screen for viewing |

Table 3.5: View the recognized word

# Chapter 4

# Design

System design defines the architecture, components, modules, interfaces, and data for system to satisfy specific requirements. Following represents the system design for developed application.

## 4.1 System Architecture

The developed system is an android application. The application has a simple architectural design comprising of following steps:

- Video as an input.
- Application of image analysis algorithm for video frames.
- Output the word after successful identification of the word.

The main physical components in the proposed system are input, processing and output. System architecture is illustrated in Figure 4.1.

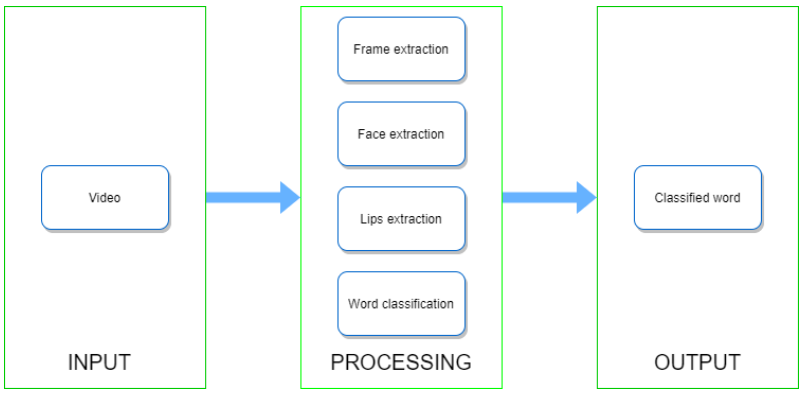


Figure 4.1: System Architecture

## 4.2 Design Constraints

One major constraint imposed on the system is the use of a small dataset for its training for word detection due to un-availability of bigger dataset for our application to train on. Due to this limitation, our application is intended to identify smaller number of words due to nature of the dataset used. There are ten words our application is expected to identify. The model used to train our system only take video as an input, but its size is limited. Thus, user is expected to input videos of smaller sizes for application to work with. This limitation is subjected to change in the future. Our application only works on android specific environment.

## 4.3 Design Methodology

The methodology to be used must follow the following steps which include image processing and classification for lip reading. A general approach is demonstrated through Figure 4.2.

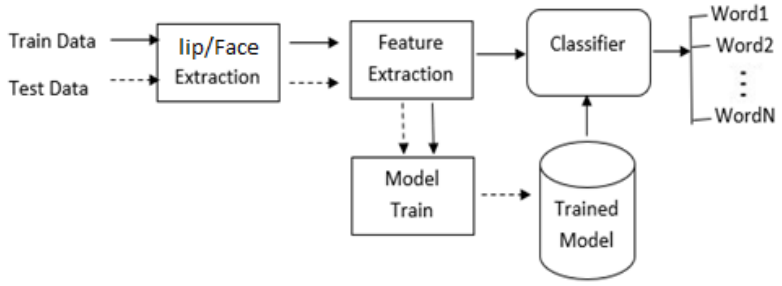


Figure 4.2: Methodology

A set of video frames belonging to same or different videos each of which represents a shape corresponding to some sound or spoken word by the subject will be loaded. Then the face or lip portion from each image/frame will be extracted. From these frames consisting of lips, features will be extracted and sent to the train model. At this point we will have a trained model. This trained model will be integrated with the application. The test data will be going through same steps till feature extraction. These features will be provided to the trained model which will then classify or recognize the word in the video. The steps involved shall be:

1. Extraction of frames from video
2. Detection of face area.
3. Face landmarks detection.
4. Extraction of lip region.
5. Classification of extracted regions for identification of words.

The dataset to be used for this purpose is MIRACL-VC1 which is available on Kaggle. It consists of ten utterances of ten different words and phrases spoken by ten females and five males. It consists of total 3000 instances and each of which consists of color and depth images.

## 4.4 High Level Design

High level designs have different viewpoints. Following are typical viewpoints:

### 4.4.1 Component Diagram

The component diagram shows the overall high level view of the application and explains the functionalities and responsibilities of the system, and how different tasks are divided and assigned to different components. Our system can be divided into three main modules as illustrated in Figure 4.3.

- Interface
- Word detection
- Word generation

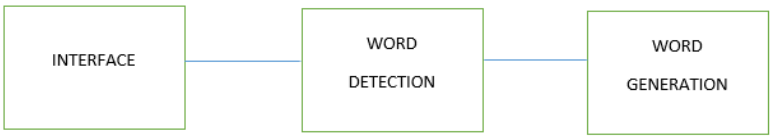


Figure 4.3: Component Diagram

### 4.4.2 System Interaction Diagram

System interaction diagram shows how the various processes within the system interact with one another. It also shows how the messages are exchanged between the objects to carry out the functionalities of the given scenario. Figure 4.4 shows the system interaction diagram.

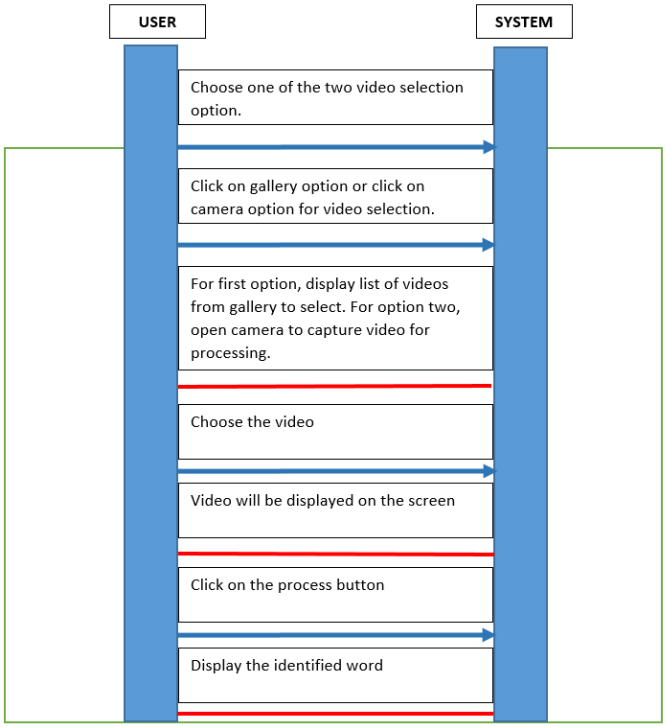


Figure 4.4: System Interaction Diagram

### 4.4.3 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Deployment diagram is displayed in Figure 4.5.

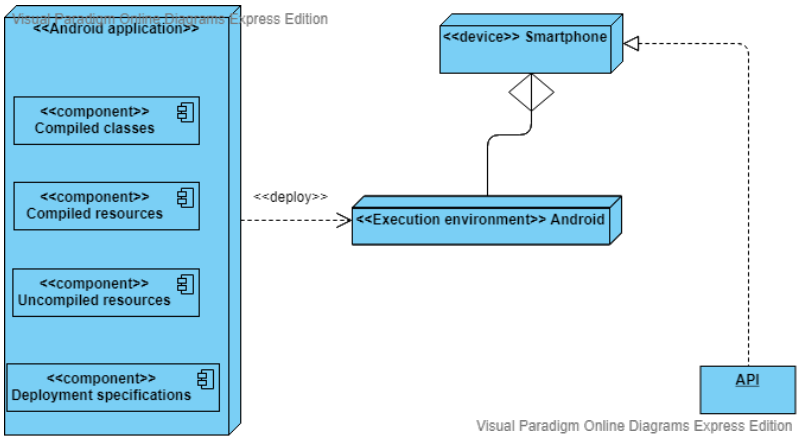


Figure 4.5: Deployment Diagram

### 4.4.4 Class Diagram

Class diagram can be referred to as the main building block of object-oriented modeling. General conceptual modeling can be taken as one of the main usages of class diagram. It helps to describe the operations and attributes of the class and constraints imposed. It shows a collection of classes, associations, collaborations, interfaces and constraints. Class diagram can be shown in Figure 4.6 below.

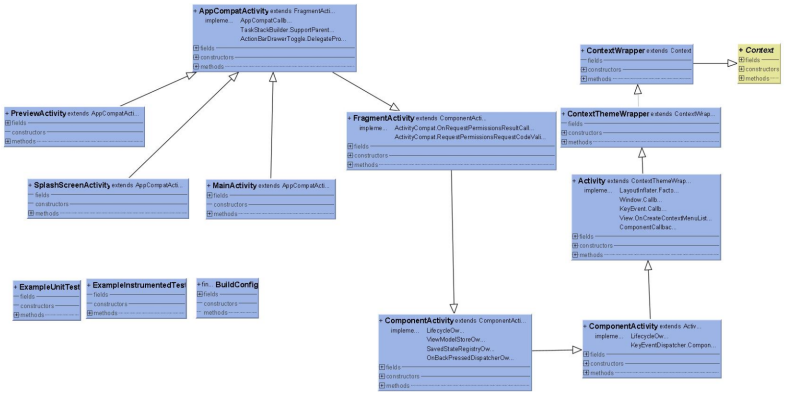


Figure 4.6: Class Diagram

## 4.5 GUI Design

The product offers easy to use and manage GUI for proper usage without having to understand any complicated technicalities. The application is a word recognition application and GUI built around it works in manner, that makes it easy for users to input their video of choosing and then get the required output in the form of word. Following points will illustrate an overview of GUI of the application:

- User will have to click an applications icon identifying our app through his/her smartphone.
- User will then be subjected to a welcome screen represented in the following Figure 4.7.
- User will now access home screen where various options regarding the functionality of our application can be accessed. Figure 4.8.
- The input is in the form of a video that can be browsed directly from the smartphone gallery using “BROWSE” button or using a built in application camera to capture the video by pressing “CAMERA” button for processing.
- After the selection of the video, user will press “PROCESS” button for processing of the video. A processing screen will be displayed till the word is displayed. Figure 4.9.
- Word will then be displayed on the separate screen and then user will be able to see it and then move back to the “Home screen” for either processing another video or exiting the application. Figure 4.10.

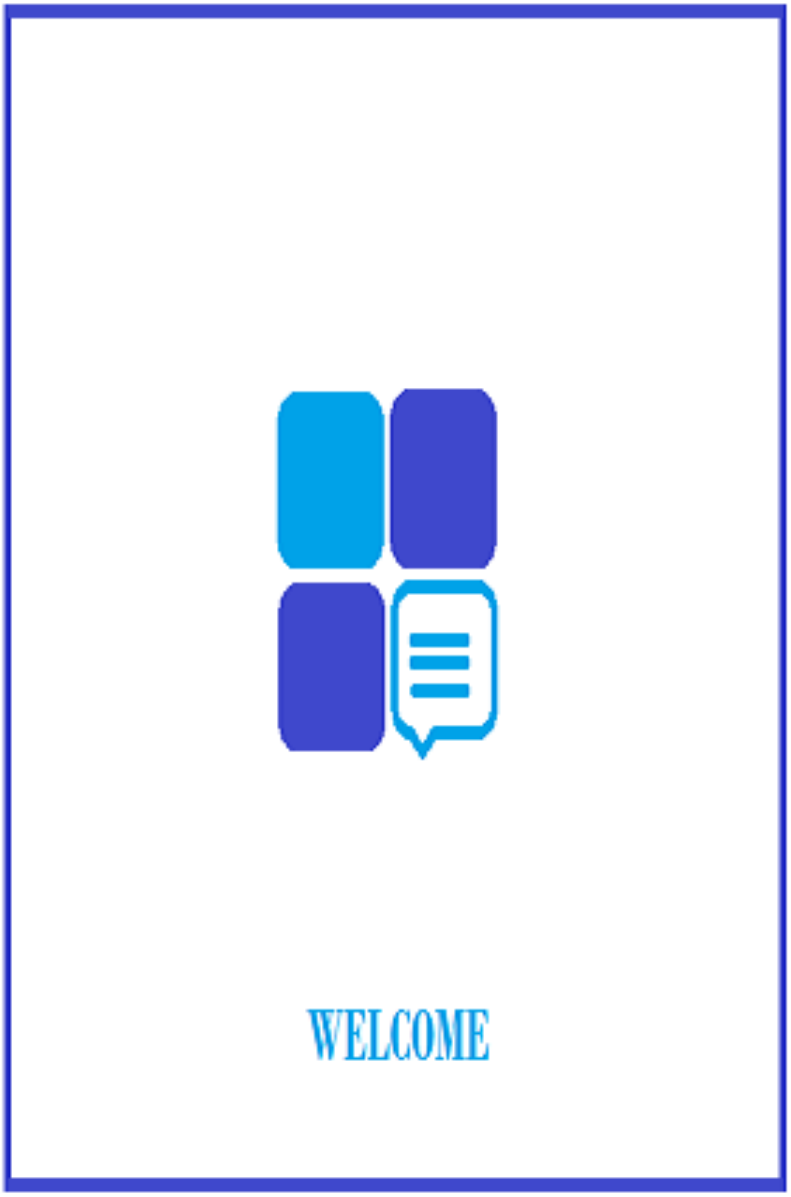


Figure 4.7: Welcome/Splash Screen

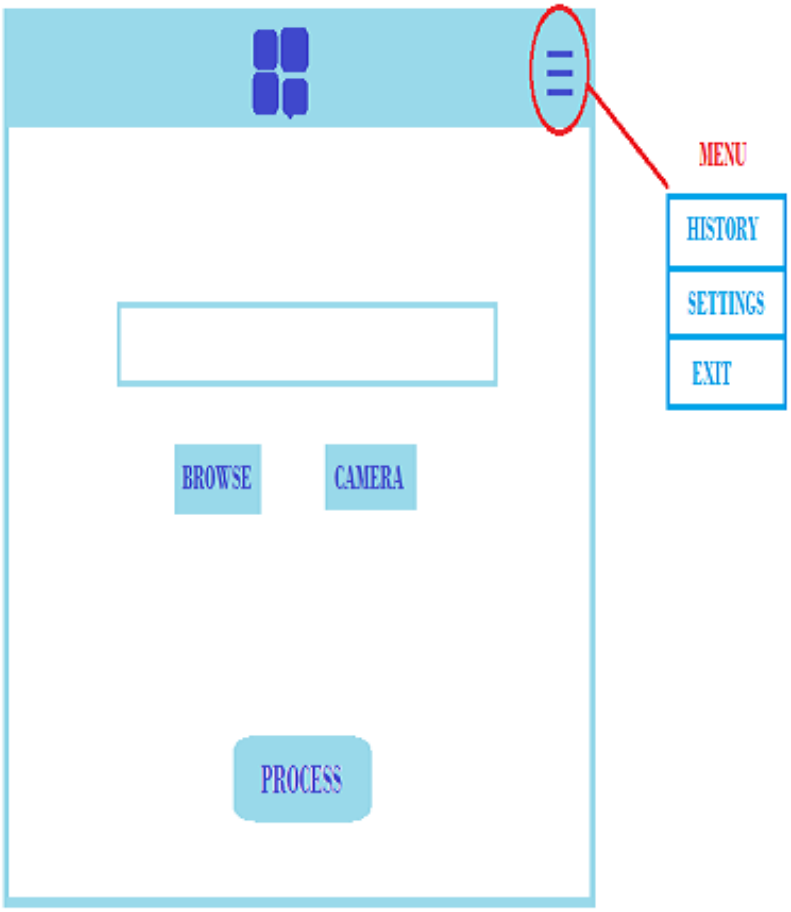


Figure 4.8: Home Screen

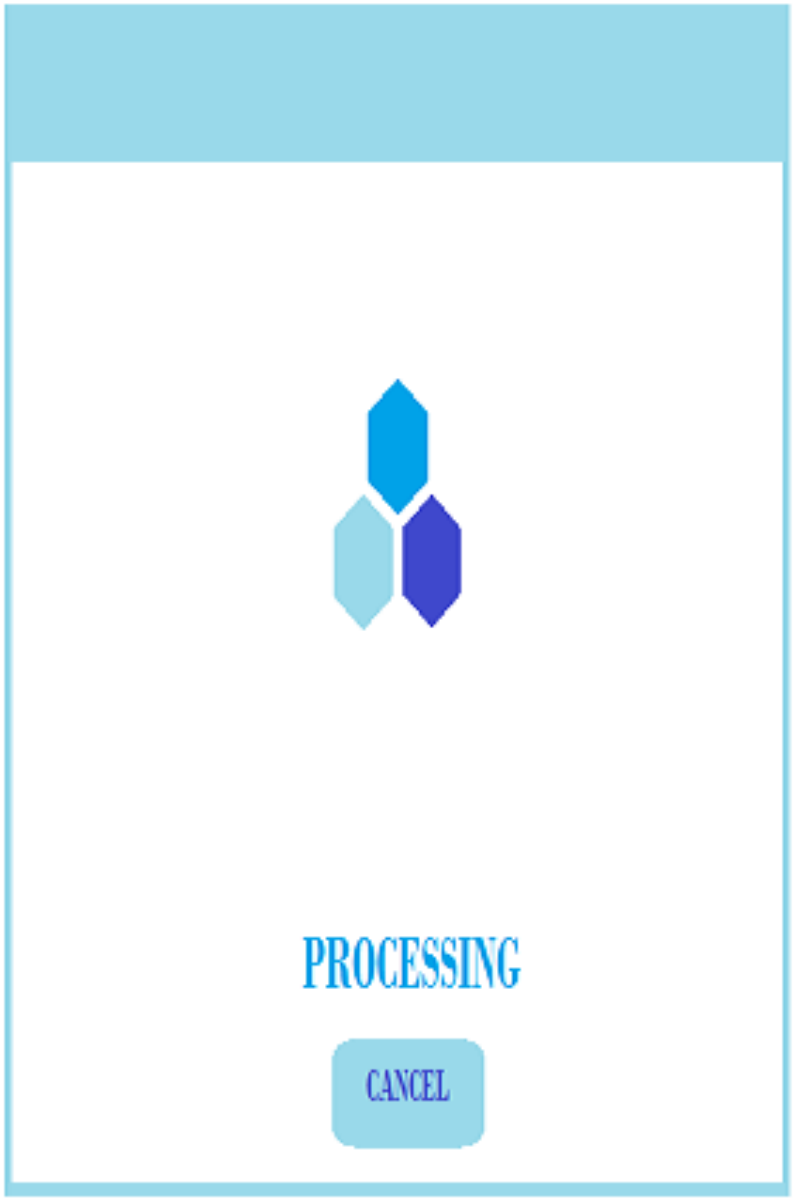


Figure 4.9: Processing Screen


Figure 4.10: Recognized word display screen

# Chapter 5

# System Implementation

## 5.1 System Architecture

The system architecture can be divided into three parts: input, processing and output. The inputs consists of reading videos. The processing part consists of various algorithms that work on the video and recognize the spoken word. This is the core part of the system which implements various algorithms such as converting video to frames, detecting face in each frame, detecting and extracting the lips region on the detected face, resizing the lips region frame and appending black frames to make the length of the video match the requirement of the deep learning model to which these frames are fed to for recognition while preserving the temporal nature of the video. Finally, the output is the word that the model has recognized.

### 5.1.1 Algorithmic Workflow

The system activity diagram showing the overall work-flow of the system is presented in Figure 5.1

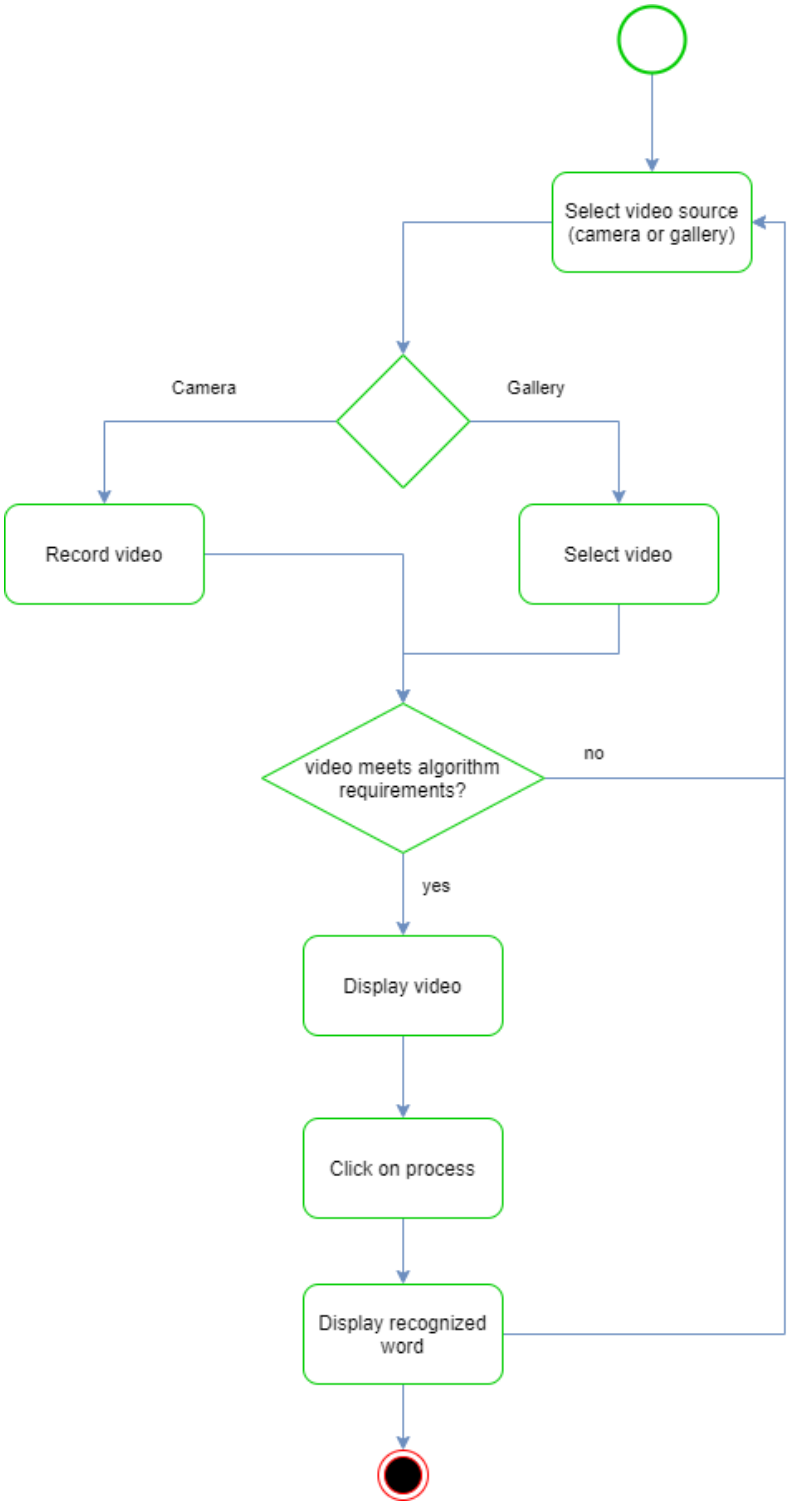


Figure 5.1: Activity Diagram of System

## 5.2 System Components

The following is a list of the components of the system based on algorithms deployed:

- Convert video to frames
- Face detection
- Face landmarks detection and lip region extraction
- Pre-processing
- Word recognition using deep learning model

### 5.2.1 Convert video to frames

The system takes video as input. From this video, the system needs to extract frames. The number of frames in the video depends upon its frame rate and the duration. The newer versions of android have made this easy as any frame of a video can simply be retrieved by its index. However, older android versions do not support this and closest frames can be retrieved through any microsecond of the video. To avoid getting duplicate frames, proper calculations have to be made based on total duration and frame rate in order to get the a list of microseconds at which unique frames can be retrieved.

### 5.2.2 Face detection

For detecting faces in each frame two separate pre-trained models were tried. The first one was based on CNN [8] and the other one was based on histogram of oriented gradients (HOG) and linear Support Vector Machine (SVM). The latter was used in the system as it is significantly faster than CNN [8] while CPU is used for processing. The performance of CNN model can be significantly enhanced if a GPU is used for its processing but GPUs in android devices are neither powerful enough nor compatible to perform such operations. Face detection displayed in Figure 2.5

### 5.2.3 Face landmarks detection and lip region extraction

Face landmarks detection can help find points of interest on a face such as eyes, nose and lips. This is also a pre-trained model that takes a frame along with the bounds of the detected face (which were obtained in the face detection phase) in it and locates 68 points of interest of the face in the frame. The points higher than 48 are points on the lips. The system creates a bounding rectangle around these points which is then used to extract this portion from the frame and hence obtain a frame which consists only of lip region of the face. This process is repeated for all frames in the video.

### 5.2.4 Pre-processing

After obtaining the lip region frames, the system pre-processes these frames to convert them into a format which is acceptable by deep learning model which recognizes which word is spoken in the video. This includes resizing the frames, appending black frames to make video of length equal to that acceptable by the model.

### 5.2.5 Word recognition using deep learning model

The system finally sends the pre-processed frames to the trained deep learning model to perform classification. The model chosen to be part of the system had achieved the highest accuracy among all tested models and all tested configurations.

Tests were conducted on a simple CNN [8] with Long Short Term Memory (LSTM) [9] layer, a deep layered CNN [8] with bi-directional LSTM [9]. In addition, some pre-defined architectures were also tested including VGG16 [11], VGG19 [10] and ResNet50 [13] with additional bi-directional LSTM [9] and fully connected layer as their top layers.

Different activation functions, kernel sizes for CNNs [8], filters for CNNs [8], units of certain layers such as LSTM [9], and learning rate were experimented with but no difference was observed in accuracy.

These experiments were performed on Miracl-VC1 as well as the data-set collected by us. To train model on more training samples, both the data-sets were combined and the chosen model was trained on it. Some models have been explained below:

#### 5.2.5.1 CNN + LSTM

Our first model used was CNN [8] +LSTM [9] which used CNN [8] layers for feature extraction and then in combination with LSTMs supported the sequence prediction, which made it many-to-one RNN. Highest probability of words from softmax activation layer was selected when it was used after adding a fully connected layer that mapped 10 units.

#### 5.2.5.2 Deep Layered CNN + LSTM

Deep Layered CNN [8] + LSTM [9] is the expansion of baseline CNN [8] + LSTM [9] thus an understanding of baseline model is necessary for this model. This model is specifically designed for sequence prediction problems with spatial inputs like images or videos. The expansion is done by adding two more CNN [8] layers for complex understanding of the image data. Bidirectional LSTM [9] was used which helped us to avoid the outweighing of the output in later sequences, and then we added dropout layer which helped us to prevent the model from overfitting. It did it by randomly setting the outgoing edges of hidden units to 0 at each update of training phase. We then also added batch normalization after every CNN [8] layer which helped us training our deep neural network which helped in standardizing the inputs to the layer for each mini-batch. This stabilized the learning process and reduced the number of epochs required to train our deep network. The dropout probability was set to 0.2 as we performed it several times across the model. We also interspersed 2x2 Max Pooling layers with strides of two between CNNs [8] which progressively reduced the spatial size of representation to reduce the amount of parameters and computation in the network. The deep layered architecture is shown in Figure 5.2.

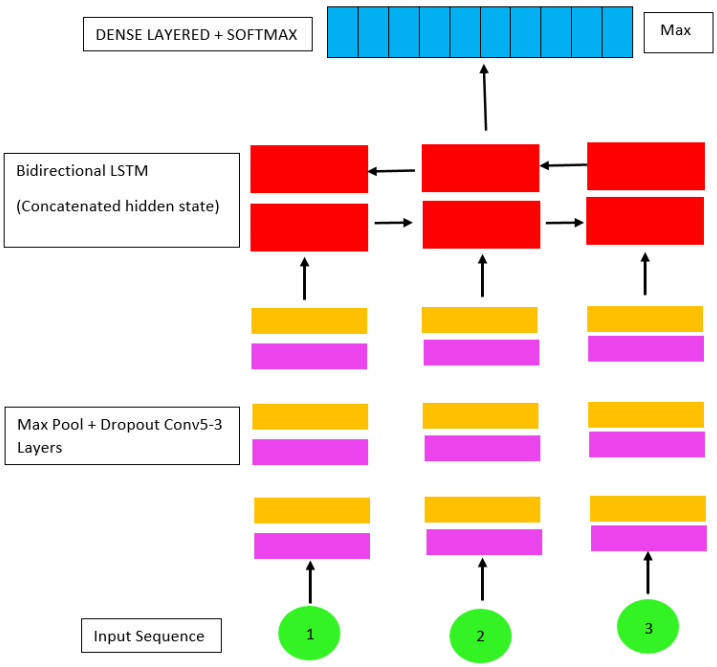


Figure 5.2: Deep Layered CNN + Bi-LSTM architecture

#### 5.2.5.3 VGG16 + BiLSTM

Due to our smaller dataset we also trained our model on VGG-16 [11] network pre-trained on ImageNet. Where smaller kernel size allowed for deeper network without increasing the parameters.

#### 5.2.5.4 VGG19 + BiLSTM

VGG 19 [10] model was also used to train our data as it uses deep convolutional neural layers and 19 layers are used on top of one another to increase the accuracy.

#### 5.2.5.5 ResNet50 + BiLSTM

ResNet 50 [13] is another architecture used to train our data. It consisted of five stages each with its convolutional and identity block. It has over 23 million trainable parameters and helped in achieving the results of our training on our dataset.

## 5.3 Tools and Technologies

The application is developed using java and Android Studio as the Integrated Development Environment (IDE). Since the application requires a lot of processing on frames which can easily be done using libraries for python, Chaquopy is employed which enables the execution of python scripts from java code. These python scripts use OpenCV-python to load frames, Dlib for face and face

landmarks detection, and keras with tensorflow backend for loading model and using it for word recognition.

# Chapter 6

# System Testing and Evaluation

## 6.1 Graphical User Interface Testing

The GUI is designed with ease of use in mind as it is an important component of our android application and provides intractability with of the application with the end user. It is simple yet attractive and does not require any outlying ingenuity to get used to. Many test subjects were allowed to interact with the application and they did not find any bug or problem, but rather found the interface to be simple and easy to use.

## 6.2 Usability Testing

Usability testing was carried out with special attention in mind as it ensures the actual usability of the system and make sure that there are no usability issues faced by the application. The developed system is very simple and only requires few user interactions. Our application was tested by ten people to look into usability issues as they had to load the video file and then process it for the word identification. During the while duration of their testing, they find the interaction with the application smooth and without any issues.

## 6.3 Software Performance Testing

Software performance testing in general is a testing practice performed to determine how the system performs in terms of responsiveness and stability under particular workload. According to various test subjects, our application is fairly responsive and smooth at times depending on the system on which it is used and it also does not take much of the memory resources to work. However, processing of videos can consume a lot of processing power and still may take considerable time depending upon the processing power of the device. Although some preliminary files start loading in separate parallel threads when the application starts, it still takes time to load. However, these

files only need to be loaded once when the application starts so the first time user inputs video for recognition, it takes more time but after the first time the recognition happens a lot faster for any video.

## 6.4 Exception Handling

There are two exceptions addressed in the application. First is the frames of the videos should be of a specific length. Frames must be less then or equal to what model accepts. Second exception suggests that faces should be extracted in 70 percent of the frames.

## 6.5 Load Testing

Load testing is the simplest form of performance testing as it is conducted to understand performance of the system under expected load. Our application uses videos to identify and load words and thus we performed load testing by giving our application videos of various lengths to process. Our application works best with videos having less than or equal to 90 frames and anything above it is not processed by the application as the model used to create our application demands that limitation. Thus video clips with large duration can cause the application to not work properly and thus hindering the experience.

## 6.6 Installation Testing

Installation testing is the check that software application is successfully installed and is working properly on the intended system. Testing was performed on various system having various versions of the android to ensure that application is installing successfully, and taking into account the resources necessary for its installation and thus installing accordingly. Testing proved that installation process was fairly smooth and test subjects did not face any issue whatsoever. Large part of it due to the flexibility the android studio provides while developing applications on it. Since the application depends on a lot of dependencies, the APK size is fairly large and may take some time to install.

## 6.7 Test Cases

The following test cases were implemented.

### 6.7.1 Application Start-up Test Case

The following Table 6.1 shows test case that ensures that system starts correctly when application is launched.

| Test Case ID | 01 | |
|---|---|---|
| Description | Test the application start | |
| Applicable for | Runs on all android systems | |
| Requirements | REQ01 | |
| Initial conditions | None | |
| Step | Task and expected result | |
| 1 | Open application | |
| 2 | Verify the application startup on multiple systems | Pass |
| 3 | Verify the application displaying its menu screen properly | Pass |

Table 6.1: Table for test case 1

### 6.7.2 Video loading Test Case

The following Table 6.2 shows test case that ensures that video loads successfully.

| Test Case ID | 02 | |
|---|---|---|
| Description | Test video loading | |
| Applicable for | Runs on all android systems | |
| Requirements | REQ02 | |
| Initial conditions | Application is started successfully | |
| Step | Task and expected result | |
| 1 | Select one of the two options for video selection provided by the application | |
| 2 | Verify that both options help you in selecting videos | Pass |
| 3 | Verify that a process button is being displayed on the screen | Pass |

Table 6.2: Table for test case 2

### 6.7.3 Word Recognition Test Case

The following Table 6.3 shows test case that ensures that word is being recognized successfully.

| Test Case ID | 03 | |
|---|---|---|
| Description | Test word detection | |
| Applicable for | Runs on all android systems | |
| Requirements | REQ03 | |
| Initial conditions | Videos were selected successfully | |
| Step | Task and expected result | |
| 1 | Select the process button displayed on the screen | |
| 2 | Verify that the video is being processed by noticing animated processing icon | Pass |
| 3 | Verify that the word is being displayed on the screen | Pass |

Table 6.3: Table for test case 3

## 6.8 Datasets

Many researcher have used various datasets for testing. However, most are publicly unavailable. For this project, we tend to test out two different datasets. One is publicly available and the other one we ourselves have created.

### 6.8.1 Miracl-VC1 Dataset

Miracl-VC1 dataset consists of videos of 15 persons. Among these 10 are females and 5 are males. Each person utters 10 different words, 10 times each. These words include: Begin, Choose, Connection, Navigation Next, Previous, Start, Stop, Hello and Web. The Miracl dataset also contains phrases but we discard them for simplicity and consider only words. Sample frames from this dataset can be seen in Figure 6.1.

Figure 6.1: Frames from Miracl-VC1 dataset

### 6.8.2 Custom Dataset

We collected videos from 10 people all of whom were males. We discarded videos of 2 persons as they were problematic. We finally had videos of 8 people. Each person uttered 10 different words, 10 times each. We choose the words to be same as Miracl-VC1 dataset which are: Begin, Choose, Connection, Navigation Next, Previous, Start, Stop, Hello and Web. Sample frames from this dataset can be seen in Figure 6.2.

Figure 6.2: Frames from custom dataset

## 6.9 Analysis and discussion of results

The evaluation metrics for our application is defined by the accuracies calculated through various models used during the course of our project. As the model with higher accuracy is selected as the base model of our application and a source of word recognition. It is important to note that there were two datasets used. One was a pre built MIRACL-VC1 dataset, while other was our custom made dataset. Accuracies were found on both these datasets. The process was carried out by first extracting faces from the frames of the videos supplied by the datasets and then extracted the mouth portion of the frames through a pre-built extraction model. The videos were split for seen and unseen data, into training, validation and testing videos . Then those frames were subjected to five models that were selected for finding accuracies. Those models were CNN [8]+LSTM [9], Deep Layered CNN [8]+ BiLSTM [9], VGG16 [11]+BiLSTM [9], VGG19 [10]+BiLSTM [9] and ResNet50 [13]+BiLSTM [9]. The following tables shows the result of our accuracies for both original and new dataset for both seen and unseen faces, and also both seen and unseen lips.

### 6.9.1 MIRACL-VC1 Dataset (Faces)

Test were conducted by giving frame sequences consisting of faces of speakers. These face images were resized to 90*90 and had 3 channels depicting they were colored frames. There were 2 configurations of tests. Seen and unseen, the results of which are discussed below:

#### 6.9.1.1 Seen tests

Seen means that the model has previously seen and been trained on videos of a person and that the test video is a new video of a person whom the model has previously seen. In this configuration, the model had seen videos of all people in the dataset but certain videos of each person was excluded from training and were added to testing set.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 10.42 | 10.67 | 11.33 |
| Deep Layered CNN + BiLSTM | 98.92 | 72.67 | 60.67 |
| VGG16 + BiLSTM | 9.00 | 10.00 | 10.00 |
| VGG19 + BiLSTM | 10.67 | 10.00 | 10.00 |
| ResNet50 + BiLSTM | 11.92 | 10.00 | 10.67 |

Table 6.4: Table of accuracies for seen faces of MIRACL-VC1

The results in Table 6.4 show that in the seen configuration, the deep layered CNN [8] with bi-directional LSTM [9] shows best results. Larger architecture models such as VGG16 [11], VGG19 [10] and ResNet50 [13] could not properly learn from the dataset. It shows that our deep layered model has the capability to learn more from this dataset.

#### 6.9.1.2 Unseen tests

In this configuration, the tests are conducted on videos of people the model has never seen before. It is important to test in this configuration as this is more close to real world scenarios.

Since the dataset contains more females (10) than males (5), it is important to separately test model results for both genders.

The table given below is for unseen male faces:

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 12.23 | 10.00 | 10.00 |
| Deep Layered CNN + BiLSTM | 27.77 | 5.00 | 16.00 |
| VGG16 + BiLSTM | 11.54 | 10.00 | 10.00 |
| VGG19 + BiLSTM | 10.08 | 10.00 | 10.00 |
| ResNet50 + BiLSTM | 9.08 | 6.00 | 10.67 |

Table 6.5: Table of accuracies for unseen male faces of MIRACL-VC1

As seen in Table 6.5, all models performed poorly on test videos of males. However, deep layered model appears to be learning a bit more than other models. One reason for the poor results may be because the males are far less than females in the dataset.

The table given below is for unseen female faces:

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 11.15 | 10.00 | 13.00 |
| Deep Layered CNN + BiLSTM | 64.38 | 24.00 | 7.00 |
| VGG16 + BiLSTM | 9.00 | 7.00 | 10.00 |
| VGG19 + BiLSTM | 9.69 | 10.00 | 10.00 |
| ResNet50 + BiLSTM | 14.31 | 6.00 | 10.67 |

Table 6.6: Table of accuracies for unseen female faces of MIRACL-VC1

The results in Figure 6.6 are surprisingly low. As the dataset consists of more females, it was expected to perform better on females. However, this was not the case. Furthermore, the deep layered model which was showing better test results than others before showed the lowest test accuracy. But its train accuracy was still the highest.

### 6.9.2 MIRACL-VC1 Dataset (Lips)

Since we are more interested in movement of lips for classification purpose, we extract lips from each frame in the dataset. Now instead of faces, we provide model with frames consisting of lips. These frames were resized to 72*28 and had 3 channels which depict that the frames were colored.

#### 6.9.2.1 Seen tests

Seen means that the model has previously seen and been trained on videos of a person and that the test video is a new video of a person whom the model has previously seen. In this configuration, the model had seen videos of all people in the dataset but certain videos of each person was excluded from training and were added to testing set.

The table below is for seen lips.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 12.58 | 9.33 | 10.00 |
| Deep Layered CNN + BiLSTM | 74.17 | 70.00 | 64.67 |
| ResNet50 + BiLSTM | 11.58 | 8.67 | 8.67 |

Table 6.7: Table of accuracies for seen lips of MIRACL-VC1

In Table 6.7, the deep layered model outperformed all other models with highest results in train, validation and test. ResNet50 [13] showed the worst results.

#### 6.9.2.2 Unseen tests

In this configuration, the tests are conducted on videos of people the model has never seen before. It is important to test in this configuration as this is more close to real world scenarios.

Since the dataset contains more females (10) than males (5), it is important to separately test model results for both genders.

The table below is for unseen female lips.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 13.38 | 10.00 | 10.00 |
| Deep Layered CNN + BiLSTM | 54.77 | 27.00 | 44.00 |
| ResNet50 + BiLSTM | 11.92 | 10.00 | 10.00 |

Table 6.8: Table of accuracies for unseen female lips of MIRACL-VC1

Table 6.8 shows that unlike the result of unseen females faces in Table 6.6, the deep layered model performed significantly better with lip frames. However, the other models, here too, struggle to learn.

The table below is for unseen male lips.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 11.54 | 14.00 | 16.00 |
| Deep Layered CNN + BiLSTM | 37.50 | 14.00 | 23.00 |
| ResNet50 + BiLSTM | 14.46 | 9.00 | 15.00 |

Table 6.9: Table of accuracies for unseen male lips of MIRACL-VC1

The result on unseen males lips (Table 6.9) is far less than the results on unseen female lips. Possible reason is that male samples are less than female samples for training. However, the deep layered model again performed much better than the other models.

It is important to note that each video consisted of 25 frames. For the models to work, number of frames should be equal in each video. There was not quite high accuracy we expected which was mostly due to the dataset used and unavailibility of a larger more robust dataset. Thus the model with higher accuracy is Deep Layered CNN [8] + BiLSTM [9] with higher overall accuracies across the board and providing a much needed basis for the application.

### 6.9.3 Custom Dataset (Face)

Test were conducted on our own dataset by giving frame sequences consisting of faces of speakers. These face images were resized to 90*90 and had 3 channels depicting they were colored frames. Due to lack of hardware resources and the nature of our custom dataset, larger models were not tested. There were 2 configurations of tests. Seen and unseen, the results of which are discussed below:

#### 6.9.3.1 Seen tests

Seen means that the model has previously seen and been trained on videos of a person and that the test video is a new video of a person whom the model has previously seen. In this configuration, the model had seen videos of all people in the dataset but certain videos of each person was excluded from training and were added to testing set.

The table below is for seen faces.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 9.22 | 10.00 | 10.00 |
| Deep Layered CNN + BiLSTM | 69.69 | 40.00 | 31.25 |

Table 6.10: Table of accuracies for seen faces of custom dataset

The results in Table 6.10 show that despite fewer training samples, the deep layered model was still able to learn to some extent.

#### 6.9.3.2 Unseen tests

In this configuration, the tests are conducted on videos of people the model has never seen before. It is important to test in this configuration as this is more close to real world scenarios.

The table below is for unseen faces.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 10.17 | 10.00 | 10.00 |
| Deep Layered CNN + BiLSTM | 16.83 | 13.00 | 12.00 |

Table 6.11: Table of accuracies for unseen faces of custom dataset

Both the models performed rather poorly (Table 6.11). Even the deep layered model could not give better results on unseen faces of the custom dataset.

### 6.9.4 Custom Dataset (Lips)

Since we are more interested in movement of lips for classification purpose, we extract lips from each frame in the dataset. Now instead of faces, we provide model with frames consisting of lips. These frames were resized to 72*28 and had 3 channels which depict that the frames were colored.

#### 6.9.4.1 Seen tests

Seen means that the model has previously seen and been trained on videos of a person and that the test video is a new video of a person whom the model has previously seen. In this configuration, the model had seen videos of all people in the dataset but certain videos of each person was excluded from training and were added to testing set.

The table below is for seen lips.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 10.00 | 11.25 | 11.25 |
| Deep Layered CNN + BiLSTM | 63.59 | 47.50 | 52.50 |
| ResNet50 + BiLSTM | 10.63 | 10.00 | 10.00 |

Table 6.12: Table of accuracies for seen lips of custom dataset

Table 6.12 shows that deep layered model has once again learned more than other models. It had performed similarly on seen lips in Miracl-VC1 dataset.

#### 6.9.4.2 Unseen tests

In this configuration, the tests are conducted on videos of people the model has never seen before. It is important to test in this configuration as this is more close to real world scenarios. The table below is for unseen lips.

| Model | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|
| CNN + LSTM | 8.17 | 10.00 | 11.00 |
| Deep Layered CNN + BiLSTM | 36.50 | 12.00 | 16.00 |
| ResNet50 + BiLSTM | 11.17 | 11.00 | 7.00 |

Table 6.13: Table of accuracies for unseen lips of custom dataset

As shown in Table 6.13, all of the models including the deep layered model performed poorly on unseen lips on our custom dataset. However, among these models the deep layered model had performed slightly better.

Although different attempts were made in order to increase accuracy such as trying out various activation functions, choosing convolution filter and kernel size, units of layers and learning rate but no significant difference was observed.

Finally, both datasets consisting of lips frames were combined and deep layered model was tested as it had performed better in earlier testing. No significant difference was observed. However, this model was trained on all combined videos and was chosen for the application as it had been trained on more train samples.

## 6.10 Comparison

In this section, Two of our models working on Miracl-VC1 dataset will be compared with a research paper [2] who used the same models on the same dataset on faces.

The comparison of results on seen faces is shown in Table 6.14.

| Model | Result of Research paper | Our Result |
|---|---|---|
| CNN + LSTM | 39% | 11% |
| Deep Layered CNN + Bi-LSTM | 25% | 60% |

Table 6.14: Comparison of Seen configuration models

The differences may be due the possibility that a different set of seen videos might have been chosen by the authors of [2].

The comparison of results on unseen faces is shown in Table 6.15.

| Model | Result of Research paper | Our Result |
|---|---|---|
| CNN + LSTM | 10% | 11% |
| Deep Layered CNN + Bi-LSTM | 10% | 11% |

Table 6.15: Comparison of Seen configuration models

On unseen faces, we get almost the same result as achieved in [2].

# Chapter 7

# Conclusions

We developed a system that is capable of identifying words spoken in a video of a person. It does it by extracting faces and then it extracts lips through dlib face detection and mouth detection libraries. From those extracted image frames, we then split them into train, test and validation images for training the models, the best of which is used by the application for generating the result. Five models were used to test the accuracies of the system and all gave varying results. In an attempt to get better results, We developed our own dataset but results did not show much improvement even if we combined the datasets. We used Android studio to develop our application and then used chaquopy SDK tool to integrate our python code in the android application and made the pre-processing and model work. The developed application is very minimalistic in design and takes the input from both camera and mobile gallery as well for processing.

## 7.1 Future Work

Due to unavailability of a larger dataset which could be much beneficial for the project, we could not get the results we were hoping for. There was one large dataset owned by BBC, but it required complicated agreement requirements in order to get hands on it, so we were not able to qualify for those requirements. But we sure did got somewhere to the point that our application was able to recognize words. Though much larger dataset would have been proved much valuable for increased accuracies.

There are aspects of the application we definitely want to enhance such as increased number of words being recognized using a larger more robust dataset for training. we would also want to add functionality to the system, that takes in unrecognized words not trained upon by the application, and then after user identification that word which was failed to recognized by the system, create new set of data for that word and then train the model on it as well. This will help a great deal in making our application to have a more real time feel. Also users will be able to add their own words and create their own datasets which can be used to train the application further. We want

to add the ability to recognize phrases and sentences by our application in future as it will surely increase the scope of our application further. Also more complicated scenarios should be addressed by our application in the future, such as multiple people in a video which can help a great deal in security and surveillance applications, thereby increasing the scope of the application.

# Appendix A

# User Manual

## Introduction

User manual is a guide that helps a user walk through an application to help him use it for the intended purpose. Following is the user guide of the application:

## A.1 Start Application

The following Figure A.1 shows start up screen of application where some preliminary files are being loaded.


Figure A.1: Loading Screen

## A.2 Video source option screen

The following Figure A.2 shows the screen where you can choose your video source to be gallery or camera.

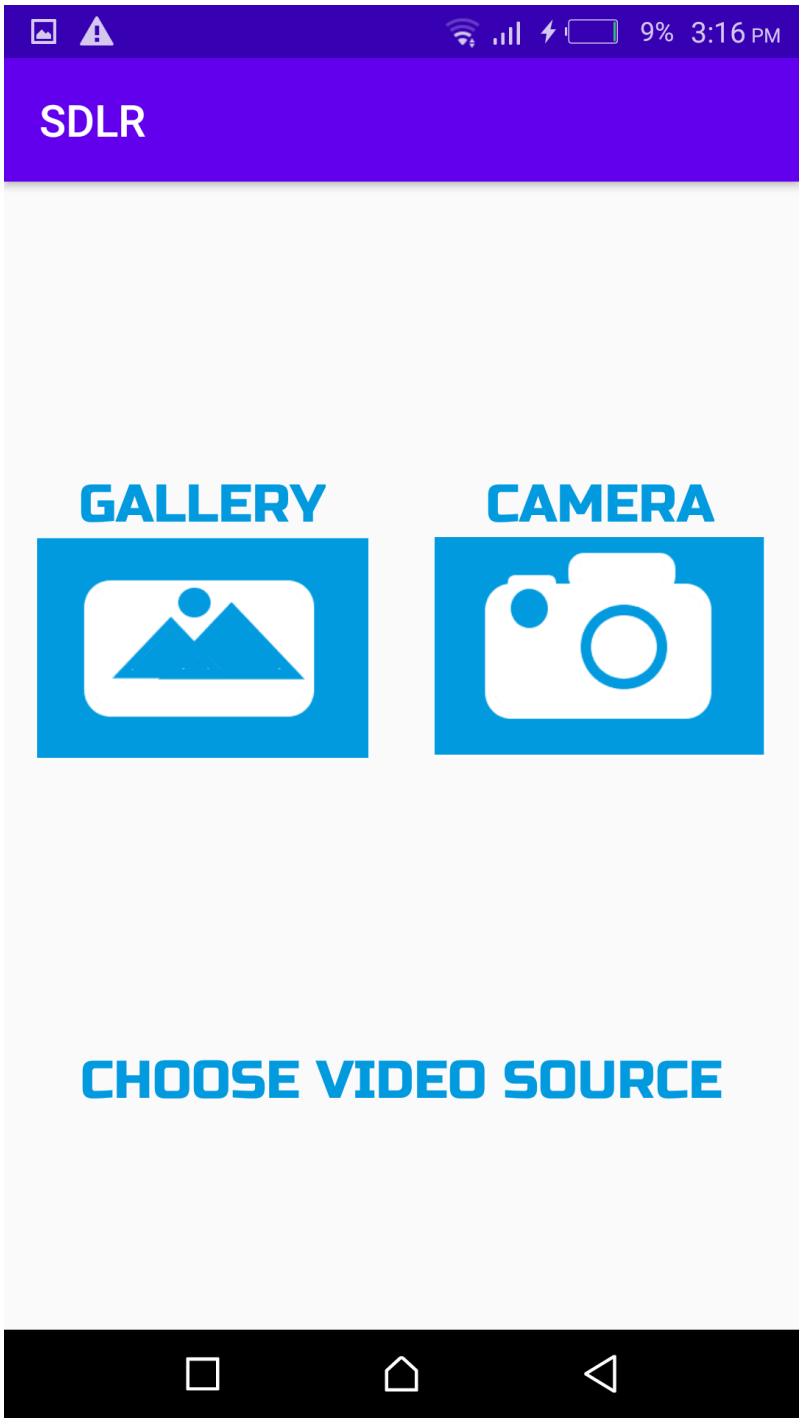


Figure A.2: Video source option screen

## A.3 Gallery

The following Figure A.3 shows how the gallery will display videos for importing into the application.

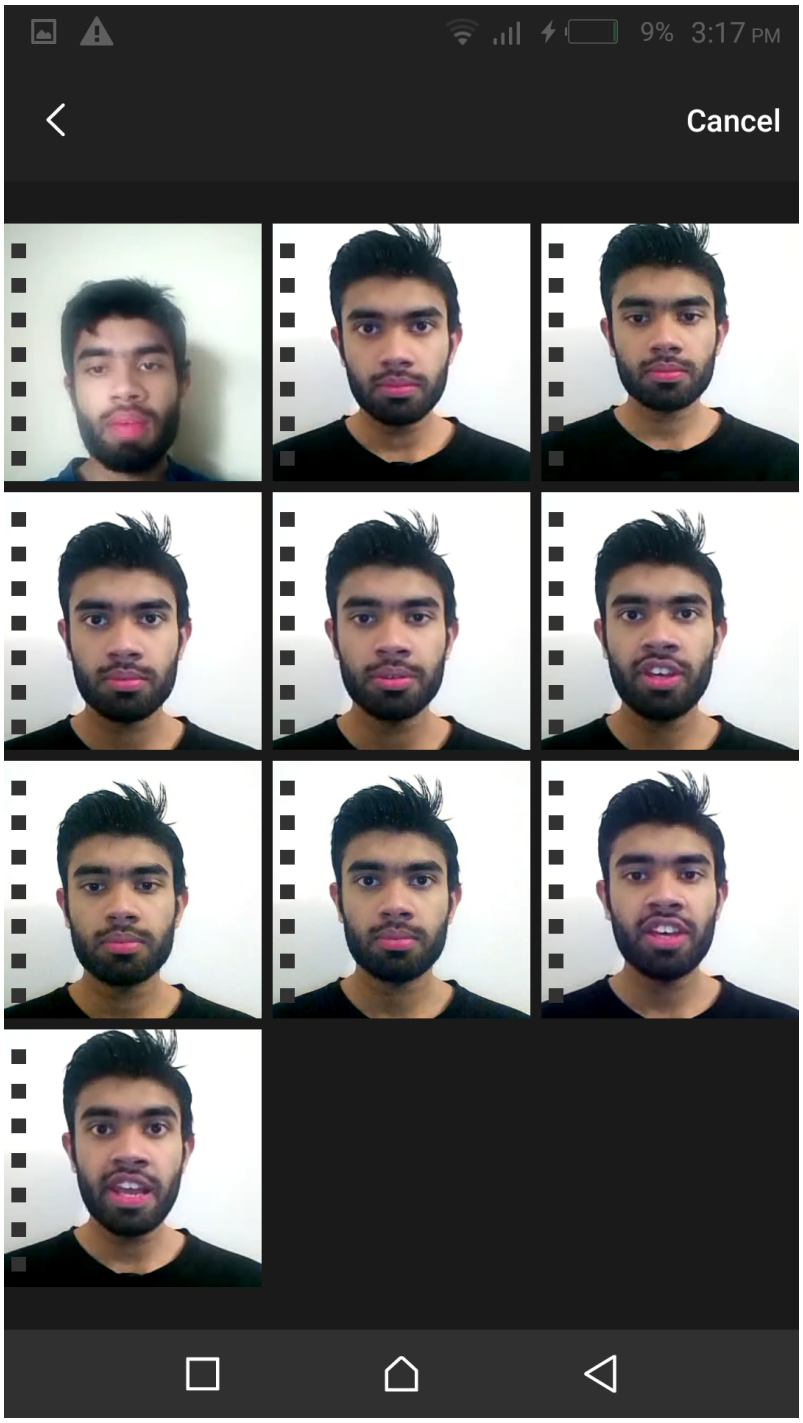


Figure A.3: Gallery for selecting video

## A.4 Preview screen

The following Figure A.4 shows how screen previews the video and allows user to proceed with processing using the process button.


Figure A.4: Preview screen

## A.5 Processing Screen

The following Figure A.5 and Figure A.6 show how once the user clicks the process button, processing will begin while user will be updated with ongoing process.

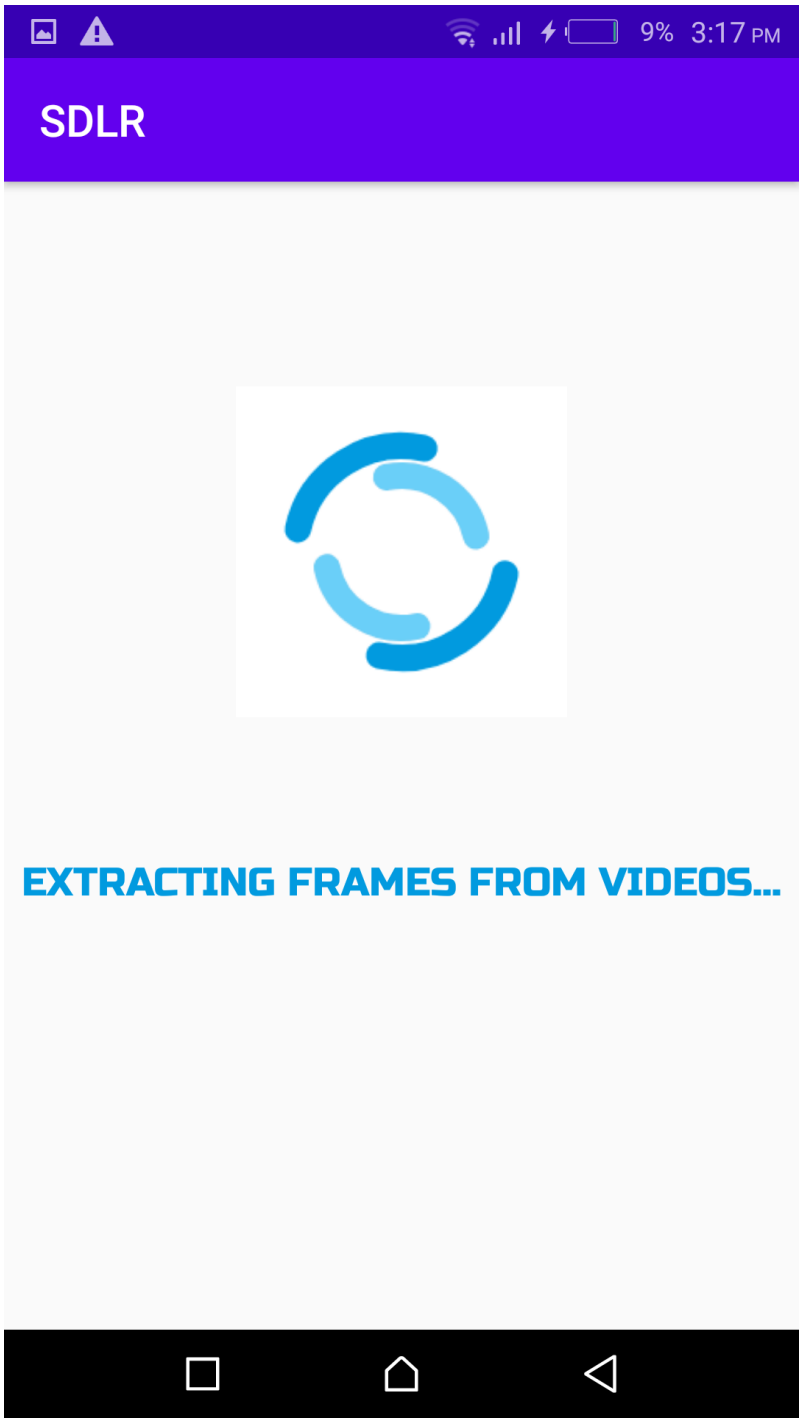


Figure A.5: Processing screen message 1

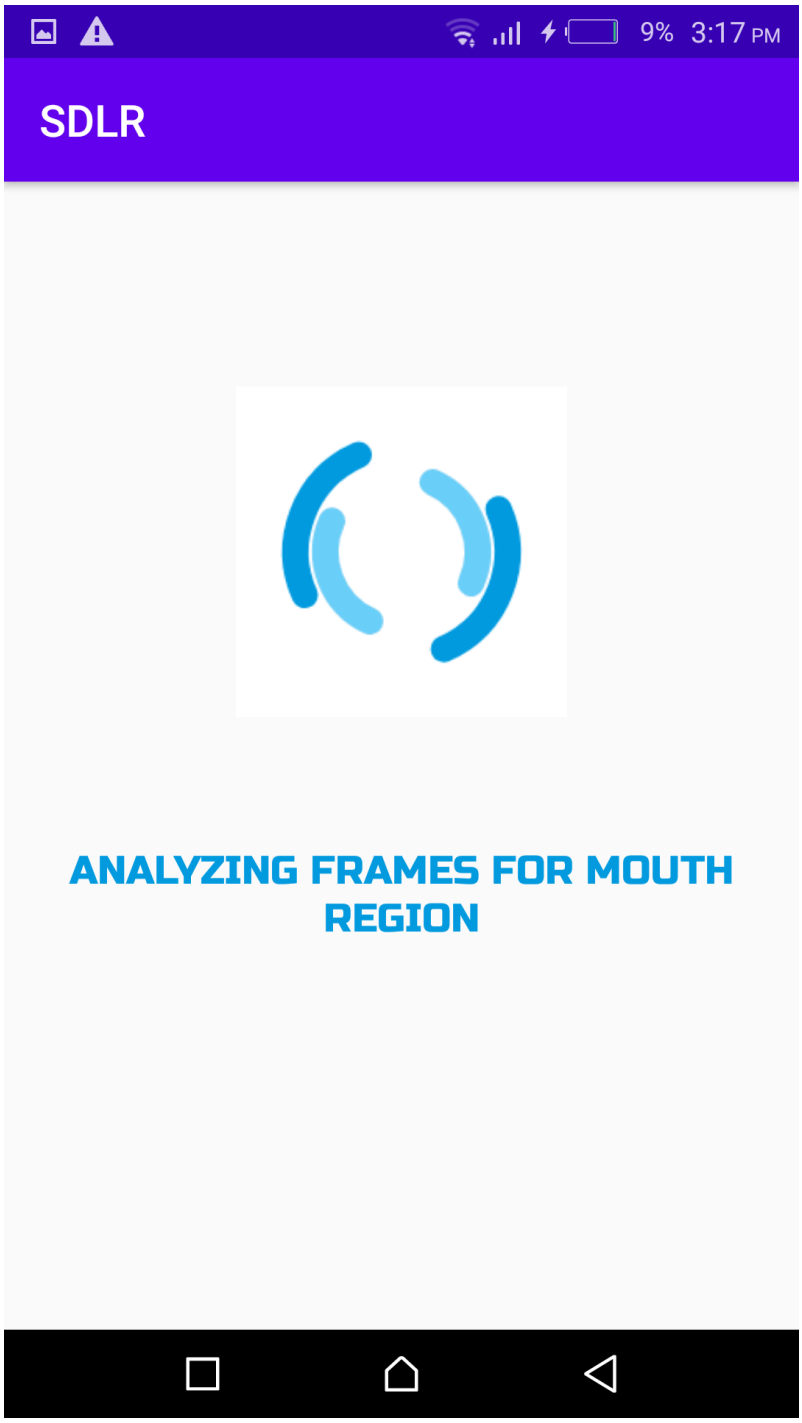


Figure A.6: Processing screen message 2

## A.6 Result screen

Once processing is done the following screen will be displayed which displays result as well as gives user to go back and select another video source. This can be seen in Figure A.7.

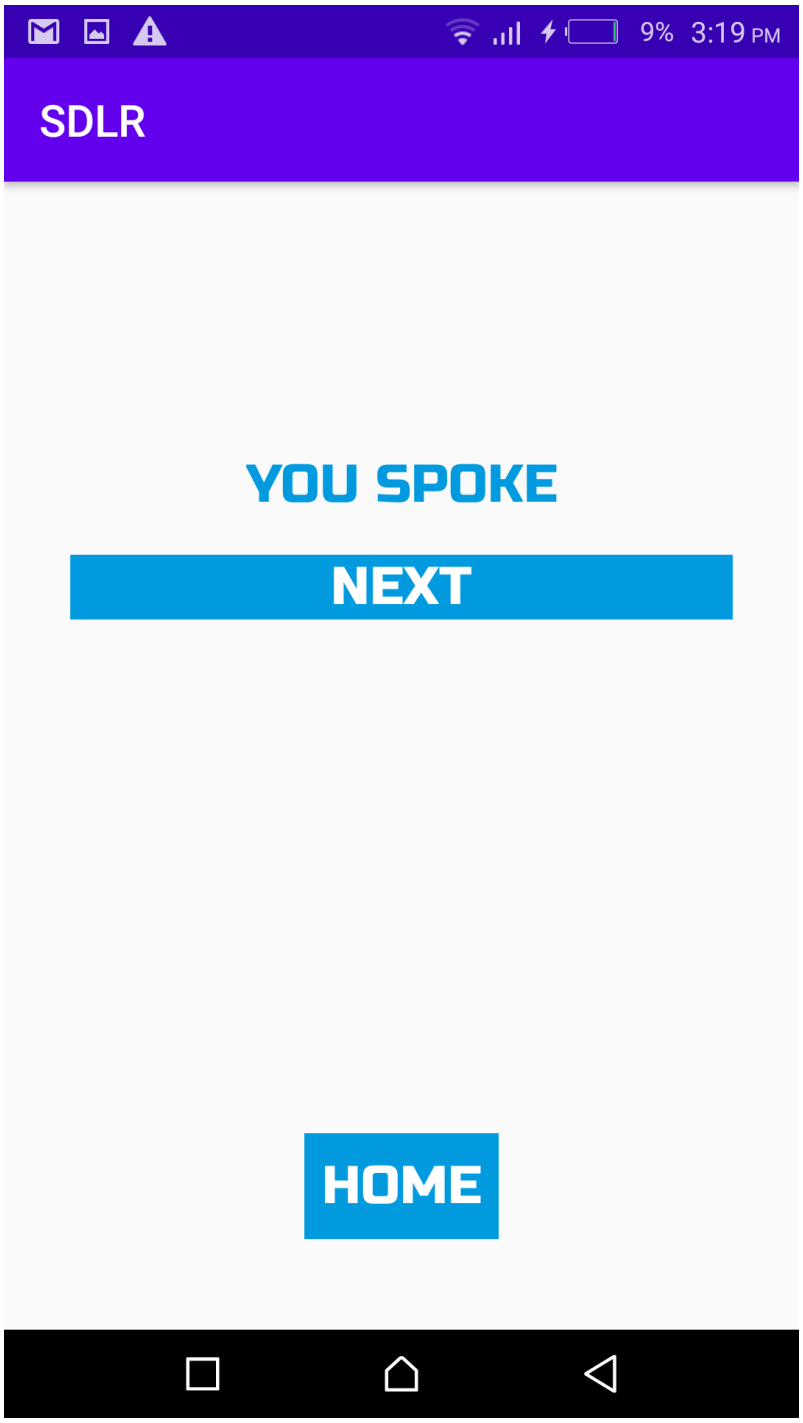


Figure A.7: Result screen

# Bibliography


[1] Yuanyao Lu and Hongbo Li. Automatic lip-reading system based on deep convolutional neural network and attention-based long short-term memory. *Applied Sciences*, 9(8):1599, 2019. Cited on pp. vii, 1, 5, 6, and 7.

[2] Abiel Gutierrez and Zoe Alanah Robert Stanford University. Lip reading word classification. Cited on pp. vii, 1, 5, 6, 7, 8, 9, 45, and 46.

[3] Joon Son Chung and Andrew Zisserman. Learning to lip read words by watching videos. *Computer Vision and Image Understanding*, 173:76–85, 2018. Cited on pp. vii, 1, 9, 10, and 11.

[4] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3444–3453. IEEE, 2017. Cited on pp. vii, 1, 10, and 11.

[5] Ahmed Rekik, Achraf Ben-Hamadou, and Walid Mahdi. A new visual speech recognition approach for rgb-d cameras. In *International Conference Image Analysis and Recognition*, pages 21–28. Springer, 2014. Cited on pp. vii, 11, and 12.

[6] Abhishek Jha, Vinay P Namboodiri, and CV Jawahar. Word spotting in silent lip videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 150–159. IEEE, 2018. Cited on p. 1.

[7] Lynn Woodhouse, Louise Hickson, and Barbara Dodd. Review of visual speech perception by hearing and hearing-impaired people: clinical implications. *International Journal of Language & Communication Disorders*, 44(3):253–270, 2009. Cited on p. 1.

[8] Qing-Bin Gao and Zheng-Zhi Wang. Center-based nearest neighbor classifier. *Pattern Recognition*, 40(1):346–349, 2007. Cited on pp. 5, 6, 7, 8, 10, 11, 13, 31, 32, 40, 41, and 43.

[9] Suman Ravuri and Andreas Stolcke. Recurrent neural network and lstm models for lexical utterance classification. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. Cited on pp. 5, 6, 7, 8, 10, 11, 13, 32, 40, 41, and 43.

[10] Manali Shaha and Meenakshi Pawar. Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660. IEEE, 2018. Cited on pp. 6, 32, 33, 40, and 41.

[11] Hussam Qassim, David Feinzimer, and Abhishek Verma. Residual squeeze vgg16. *arXiv preprint arXiv:1705.03004*, 2017. Cited on pp. 8, 13, 32, 33, 40, and 41.

[12] Daehyun Lee, Jongmin Lee, and Kee-Eung Kim. Multi-view automatic lip-reading using neural network. In *Asian conference on computer vision*, pages 290–302. Springer, 2016. Cited on p. 11.

[13] Ivan Gruber, Miroslav Hlaváč, Miloš Železnỳ, and Alexey Karpov. Facing face recognition with resnet: round one. In *International Conference on Interactive Collaborative Robotics*, pages 67–74. Springer, 2017. Cited on pp. 32, 33, 40, 41, and 42.

# References


[1] Yuanyao Lu and Hongbo Li. Automatic lip-reading system based on deep convolutional neural network and attention-based long short-term memory. *Applied Sciences*, 9(8):1599, 2019. Cited on pp. vii, 1, 5, 6, and 7.

[2] Abiel Gutierrez and Zoe Alanah Robert Stanford University. Lip reading word classification. Cited on pp. vii, 1, 5, 6, 7, 8, 9, 45, and 46.

[3] Joon Son Chung and Andrew Zisserman. Learning to lip read words by watching videos. *Computer Vision and Image Understanding*, 173:76–85, 2018. Cited on pp. vii, 1, 9, 10, and 11.

[4] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3444–3453. IEEE, 2017. Cited on pp. vii, 1, 10, and 11.

[5] Ahmed Rekik, Achraf Ben-Hamadou, and Walid Mahdi. A new visual speech recognition approach for rgb-d cameras. In *International Conference Image Analysis and Recognition*, pages 21–28. Springer, 2014. Cited on pp. vii, 11, and 12.

[6] Abhishek Jha, Vinay P Namboodiri, and CV Jawahar. Word spotting in silent lip videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 150–159. IEEE, 2018. Cited on p. 1.

[7] Lynn Woodhouse, Louise Hickson, and Barbara Dodd. Review of visual speech perception by hearing and hearing-impaired people: clinical implications. *International Journal of Language & Communication Disorders*, 44(3):253–270, 2009. Cited on p. 1.

[8] Qing-Bin Gao and Zheng-Zhi Wang. Center-based nearest neighbor classifier. *Pattern Recognition*, 40(1):346–349, 2007. Cited on pp. 5, 6, 7, 8, 10, 11, 13, 31, 32, 40, 41, and 43.

[9] Suman Ravuri and Andreas Stolcke. Recurrent neural network and lstm models for lexical utterance classification. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. Cited on pp. 5, 6, 7, 8, 10, 11, 13, 32, 40, 41, and 43.

[10] Manali Shaha and Meenakshi Pawar. Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660. IEEE, 2018. Cited on pp. 6, 32, 33, 40, and 41.

[11] Hussam Qassim, David Feinzimer, and Abhishek Verma. Residual squeeze vgg16. *arXiv preprint arXiv:1705.03004*, 2017. Cited on pp. 8, 13, 32, 33, 40, and 41.

[12] Daehyun Lee, Jongmin Lee, and Kee-Eung Kim. Multi-view automatic lip-reading using neural network. In *Asian conference on computer vision*, pages 290–302. Springer, 2016. Cited on p. 11.

[13] Ivan Gruber, Miroslav Hlaváč, Miloš Železnỳ, and Alexey Karpov. Facing face recognition with resnet: round one. In *International Conference on Interactive Collaborative Robotics*, pages 67–74. Springer, 2017. Cited on pp. 32, 33, 40, 41, and 42.