AUTOMATED LIP READING USING DEEP LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

SHYAM KRISHNAN R (CHN17CS103)

ANGEL MARY ALEX (CHN18CS017)

SNEHA ESTHER CHERIAN E (CHN18CS112)

to

the A P J Abdul Kalam Technological University

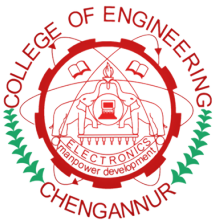
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering



DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ENGINEERING CHENGANNUR, Kerala - 689121

Phone: (0479) 2454125, 2451424; Fax: (0479) 2451424

JUNE, 2022

DECLARATION

We undersigned hereby declare that the project report "AUTOMATED LIP READING USING DEEP LEARNING TECHNIQUES", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Mrs Betty James. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Chengannur

Date: 27th June 2022

SHYAM KRISHNAN R

ANGEL MARY ALEX

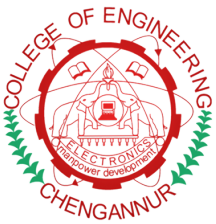
SNEHA ESTHER CHERIAN E

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ENGINEERING CHENGANNUR

KERALA

2021 - 2022



CERTIFICATE

This is to certify that the report entitled **Automated Lip Reading Using Deep Learning Techniques** submitted by **Shyam Krishnan R, Angel Mary Alex, Sneha Esther Cherian E**, to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide

Mrs. Betty James
Assistant Professor
Department of Computer Engineering
College of Engineering Chengannur

Project Coordinator

Mrs. Shiny B
Assistant Professor
Department of Computer Engineering
College of Engineering Chengannur

Head of the Department

Dr. Manju S. Nair
Associate Professor
Department of Computer Engineering
College of Engineering Chengannur

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude and sincere thanks to all who helped us to complete the work successfully. Our first and foremost thanks goes to God Almighty who showered in immense blessings on our effort.

We wish to express our sincere thanks to **Dr. Smitha Dharan. Principal College of Engineering Chengannur** for providing us with all the necessary facilities and support. We would like to express our sincere gratitude to **Dr. Manju S Nair(Head of the Department)**, for her support and co-operation. We wish to express our sincere gratitude towards all the teaching and non-teaching staff members of our Department.

Now we extend our sincere thanks to our project co-ordinator **Mrs. Shiny B**, Assistant Professor, Dept. of Computer Engineering, and our guide **Mrs. Betty James**, Assistant Professor, Dept. of Computer Engineering for guiding us in our work and providing timely advice and valuable suggestions. We also extend our gratefullness towards all the teaching and non teaching staff members of our Department.

Finally We thank our parents, all our friends, near and dear ones who directly and indirectly contributed to the success of our project.

SHYAM KRISHNAN R

ANGEL MARY ALEX

SNEHA ESTHER CHERIAN E

ABSTRACT

Lip reading is a way of using skills and knowledge to understand a speaker's verbal communication by visually interpreting the lip movements of the people. This becomes a tedious task when there are obstructions or background noise in the data. The task of automated lip-reading has attracted a lot of research attention in recent years and many breakthroughs have been made in the area with a variety of machine learning-based approaches having been implemented. But it had not been easy to create a predictive lip reading model with good accuracy mainly because most of the languages have a wide range of vocabulary and it is very much possible that the model may encounter an instance that it has not seen and identified as a particular class label.

This project focuses on building a lip-reading system by using lip movements in the video and conversion of recognized lip movements and related features into possible words. The proposed lip-reading system uses GRU based neural network to create the word detection model and training is performed on the MIRACL-VC1 dataset. This system can help people with hearing impairment understand what the other person is speaking.

Contents

List of Figures

List of Abbreviations

ASCII American Standard Code for Information Interchange

ASR Automatic Speech Recognition

BGRU Bidirectional Gated Recurrent Unit

CLAHE Contrast Limited Adaptive Histogram Equalization

CNN Convolutional Neural Network

CTC Connectionist Temporal Classification

GAN Generative Adversial Networks

GRU Gated Recurrent Unit

HOG Histogram of Oriented Gradients

RGB Red Green Blue

ROI Region Of Interest

SURF Speeded Up Robust Features

SVM Support Vector Machine

TCN Temporal Convolutional Networks

Chapter 1

INTRODUCTION

1.1 General Background

Communication is fundamental to the existence and survival of humans to express their ideas and feelings to reach a common understanding among the people. Speech is the most common form of communication between humans and involves the perception of both acoustic and visual information. Vision plays a crucial role in speech understanding. Although acoustic information is richer than visual information when speaking, most people rely on watching lip movements to fully understand speech. Furthermore, people rely on visual information in noisy environments where receiving auditory information is challenging. Similarly, people with hearing impairments depend on visual information to perceive spoken words. However, comprehending oral language using visual information alone, especially in the absence of context, can be challenging because it is difficult to understand lipreading actions such as lip, tongue, and teeth movements without context.

The task of automated lip-reading has attracted a lot of research attention in recent years and many breakthroughs have been made in the area with a variety of machine learning-based approaches having been implemented. Lipreading is the task of understanding speech by analyzing the movement of lips. Automated lip reading can be done both with and without the assistance of audio and when performed without the presence

of audio, it is often referred to as visual speech recognition. A machine that can read lip movement has great practicality in numerous applications such as automated lip reading of speakers with damaged vocal tracts, biometric person identification, multi-talker simultaneous speech decoding, silent-movie processing, and improvement of audio-visual speech recognition in general. Therefore, developing a speech recognition system that exclusively uses visual information is important.

Lipreading and audio-visual speech recognition, in general, was revolutionized by deep learning and the availability of large datasets for training deep neural networks. Lipreading is an inherently supervised problem in machine learning and more specifically a classification task. The most recent approaches to automated lip-reading largely focus on decoding long speech segments in the form of words and sentences using either words or ASCII characters as the classes to recognize. In the first case, a lipreading network receives a video where a single word is spoken and predicts a word label from the vocabulary of the dataset. In the second case, the input video may contain a full sentence (multiple words) and a deep neural network outputs a sequence of characters, which is the predicted text given the input sentence [7].

1.1.1 A general framework for automated Lip reading

Figure 1.1: Stages of automated lip reading system

Lip-reading systems typically follow a framework where there is a frontend for feature extraction, a backend for classification and some preprocessing at the start[6]. The stages are outlined in Fig 1.1:

- Visual Input - Videos of people speaking are sampled into image frames representing speech to be decoded.

- Pre-processing - This is where the region of interest (ROI), i.e., the lips are located and extracted from the raw image data. This involves detecting the face, locating the lips and extracting the lip region from the video image.

- Feature Extraction (Frontend) - This involves extracting effective and relevant features from redundant features and the mapping of high-dimensional image data into a lower-dimensional representation.

- Classification (Backend) - This involves ascribing speech to facial movements that have been transformed into a lower-dimensional feature vector.

- Decoded Speech - Speech is decoded in classes or units and eventually encoded as spoken words or sentences.

1.2 Objectives

Obtaining inspiration from existing deep lipreading networks, this project aims to develop a lip-reading system using GRU based neural network that can help people with hearing impairment understand what the other person is speaking. The model is trained on the MIRACL-VC1 dataset which contains 15 speakers uttering 10 times a set of 10 words and 10 phrases. The automated lip reading can be done in four stages: firstly, the input video is converted into frames where face detection and lip portion cropping are done. Secondly, feature extraction is done on the lip region. Thirdly, the deep learning model is created using Gated Recurrent Unit architecture and is trained on the dataset using the extracted features. Finally, the prediction of words is done using the model.

Chapter 2

LITERATURE SURVEY

2.1 LIPNET: End-to-End Sentence-Level Lipreading

In this paper[1], deep learning technology is used to perform sentence prediction. Traditional approaches separated the lipreading problem into two stages: designing or learning visual features, and prediction. However, existing work on models trained end-to-end perform only word classification, rather than sentence-level sequence prediction.

The proposed model is LipNet, the first model to apply deep learning to end-to-end learning of a model that maps a sequence of image frames of a speaker's mouth to entire sentences. The end-to-end model eliminates the need to segment videos into words before predicting a sentence. LipNet requires neither hand-engineered spatiotemporal visual features nor a separately-trained sequence model classification loss (CTC).

LipNet is a neural network architecture for lipreading that performs sentence-level sequence prediction for visual speech recognition. It takes as input a sequence of images and outputs a distribution over sequences of tokens. It is trained end-to-end using CTC(connectionist temporal classification loss) for speech recognition. The dataset used in this paper is the GRID corpus through which LipNet achieves 95.2% accuracy at the sentence level.

2.2 LCANet: End-to-End Lipreading with Cascaded Attention-CTC

This paper proposes LCANet[14], an end-to-end deep neural network-based lipreading system, which relies purely on visual information. LCANet encodes input video frames using a stacked 3D convolutional neural network (CNN), highway network, and bidirectional GRU network.LCANet introduced a cascaded attention-CTC decoder which effectively compensates for the defect of the conditional independence assumption of the CTC approach.

The proposed end-to-end lipreading system includes three major steps. (1) The mouth regions are cropped from the input video frames from aligned faces. (2) An encoder extracts Spatio-temporal features from the input sequence. (3) The cascaded attention-CTC decoder generates text from encoded hidden (Spatio-temporal) features.

LCANet uses a cascaded attention-CTC decoder whose benefits are it avoids tuning the weight parameter in the loss function and it generates the output directly from the softmax output instead of merging the results from both CTC and attention branches. The dataset used in this paper is the GRID corpus. The LCANet is implemented and trained using Tensorflow.

2.3 Text Extraction through Video Lip Reading Using Deep Learning

This paper[4] proposed a method of converting video data to text data through lip reading. The proposed method includes a test dataset, image frame analysis and having text output from identified words. In the proposed technique, the test dataset will be organized by combining all the possible facial expressions of different words.

The proposed system can be divided into four main sections. They are – test data, data preprocessing, and identification of word and output. In the test data section, raw video clips will be considered as input for the system. This data will be preprocessed in the data preprocessing section and data will be split into frames. Those frames are the processed data and will be matched with training data in the identification of the word section. Determinately the system will provide results predicated on the analysis in the output section.

2.4 Lip reading using temporal convolutional networks

B. Martinez et al.[9]proposed a system to improve the performance of the state-of-the-art model for recognition of isolated words in the wild that consists of a residual network and Bidirectional Gated Recurrent Unit (BGRU) layers. In this work, they address the limitations of this model and propose changes that further improve its performance. Firstly, the BGRU layers are replaced with Temporal Convolutional Networks (TCN). Secondly, the training procedure is simplified, which allows the training of the model in one single stage. Thirdly, they show that the current state-of-the-art methodology produces models that do not generalize well to variations on the sequence length, and addresses this issue by proposing a variable-length augmentation. The results are presented on the largest publicly available datasets for isolated word recognition in English and Mandarin, LRW and LRW1000, respectively. The proposed model results in an absolute improvement of 1.2% and 3.2%, respectively, in these datasets which is the new state-of-the-art performance.

2.5 Improving viseme recognition using GAN-based frontal view mapping

Mattos et. al.[11] proposed a novel methodology to tackle the problem of recognizing visemes – the visual equivalent of phonemes – using a Generative Adversarial Networks (GAN) to artificially lock the face view into a perfect frontal view, reducing the view angle variability and simplifying the recognition task performed by our classification CNN. The GAN is trained using a large-scale synthetic 2D dataset based on realistic 3D facial models, automatically labelled for different visemes, mapping a slightly random view to a perfect frontal view. The method is evaluated using the GRID corpus, which was processed to extract viseme images and their corresponding synthetic frontal views to be further classified by the CNN model. Their results demonstrate that the additional synthetic frontal view is able to improve accuracy by 5.9% when compared with classification using the original image only.

Chapter 3

PROBLEM DEFINITION

Over 5% of the world's population - or 430 million people - have some form of hearing disability. According to WHO, it is estimated that by 2050 over 700 million people – or one in every ten people – will have disabling hearing loss. Automatic Speech Recognition (ASR) using only acoustic data depends heavily on the quality of the sound. If the data is polluted with noise or other disturbances an ASR system will have a much more difficult task than if the data was in perfect condition. One technique for making ASR more robust is using lip reading.

In viseme-based lip reading systems[7], the problems faced are there is no standard set for visemes and identification of viseme is difficult as they cover only a small subspace of the mouth motion[8]. So this project uses a generic framework to recognise words without resorting to viseme classification. The proposed system can serve as a hearing aid for the hearing impaired particularly interacting with people with no knowledge of sign language.

Chapter 4

PROPOSED SYSTEM

4.1 System Design

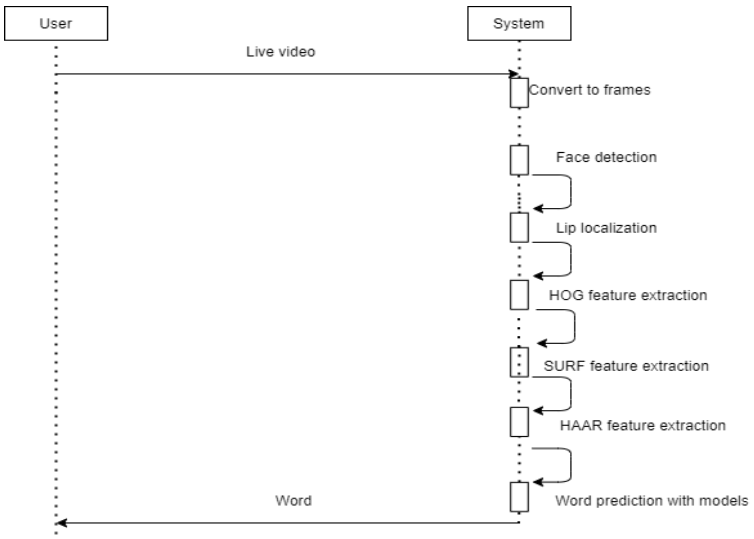


Figure 4.1: System design of proposed lip reading system

The proposed system is a GRU(Gated Recurrent Unit) based Architecture. First, the video where the speaker utters a word is given as input by the user and the frames are extracted from it. Face detection and localized lip region cropping are performed on the frames. The cropped images are passed into the visual feature extraction unit where image processing

techniques like HOG, SURF and Haar are applied to the input frames. Then training of the GRU model on three features is done them separately and the results from the neural networks are combined to make a prediction. The word prediction unit predicts the word based on the outputs of the three GRU units.

Chapter 5

MATERIALS AND METHODOLOGY

5.1 Dataset

5.1.1 MIRACL-VC1

MIRACLE-VC1 dataset [13] that consists of both depth and colour images are used for visual speech recognition, face detection and biometrics. This dataset consists a total of fifteen speakers of which five are men and ten are women who are positioned in the frustum of a MS-KINECT sensor.

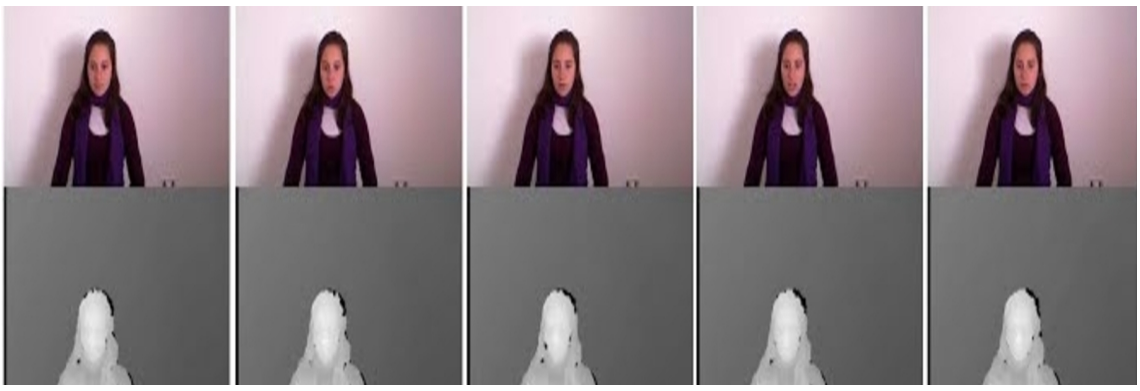


Figure 5.1: Colour and Depth images in the MIRACL-VC1 dataset

The speakers utter 10 times a set of 10 words and 10 phrases. So, it contains a total number of 3000 instances as a whole. Each instance of the dataset consists of a synchronized sequence of depth and colour images (640*480 pixels) collected at 15 frames per second. The lengths of these sequences range from 4 to 27 image frames. In our project, only colour images are used for word detection.

ID	Words		ID	Phrases
1	*Begin*		1	*Stop navigation.*
2	*Choose*		2	*Excuse me.*
3	*Connection*		3	*I am sorry.*
4	*Navigation*		4	*Thank you.*
5	*Next*		5	*Good bye.*
6	*Previous*		6	*I love this game.*
7	*Start*		7	*Nice to meet you.*
8	*Stop*		8	*You are welcome.*
9	*Hello*		9	*How are you?*
10	*Web*		10	*Have a good time.*

Figure 5.2: Words and Phrases in the MIRACL-VC1 dataset

5.2 Methodology

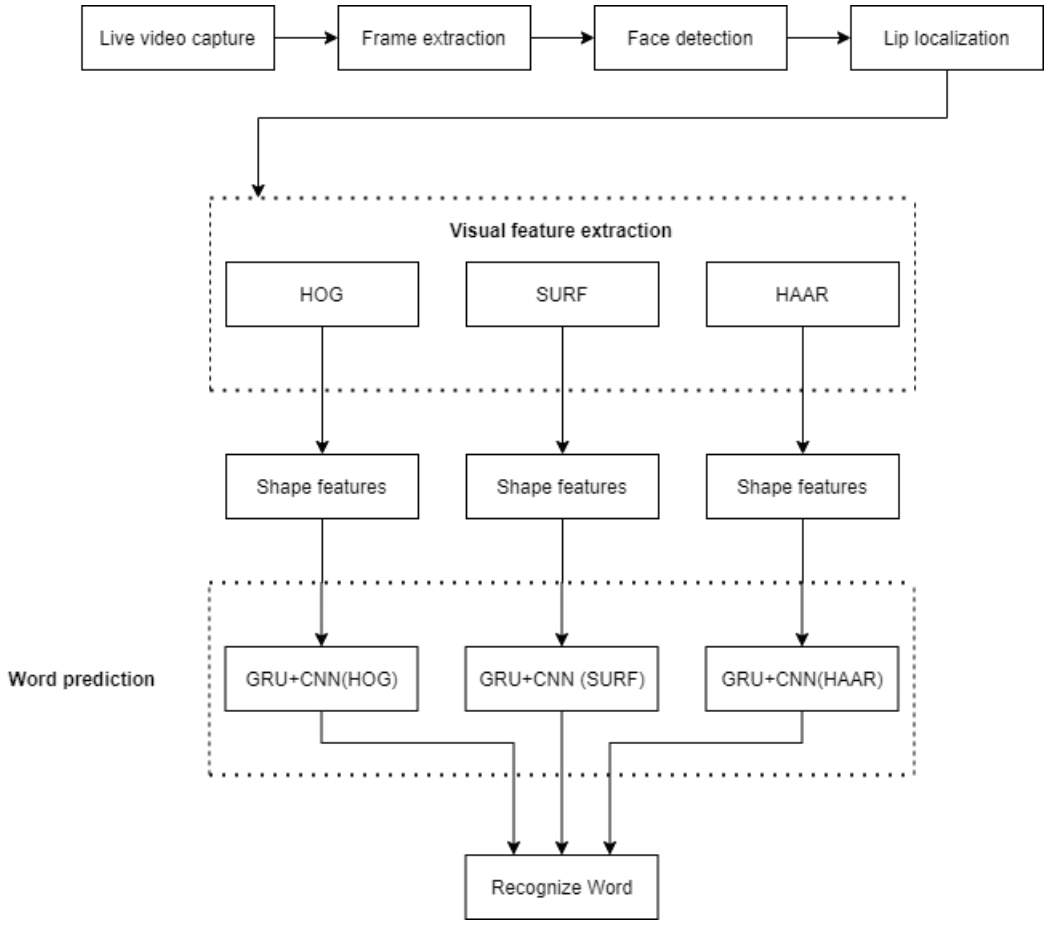


Figure 5.3: The architecture of the lip reading system

5.2.1 Video capture

The lip reading system starts with video acquisition. In this study, we used the MIRACL-VC1 dataset which consists of each speaker's utterances stored in separate folders[12]. At the beginning of the project, we need to set the path of the dataset so the system can load the file.

5.2.2 Preprocessing

Image processing is a method of performing an operation on an image to improve image quality and extract information from the image. Nowadays, image processing has become one of the common technologies that are growing rapidly in the area such as healthcare, agriculture, computer science and engineering. There are major steps in image processing where images are imported using image acquisition tools, and images are analyzed and manipulated.

(i) **Frame extraction**

This is the image acquisition stage where the frames are acquired from the MIRACL-VC1 dataset. Image processing techniques then applied on to the frames are Gaussian filter for noise removal and Clahe enhancement.

Gaussian filter - Noise is always present in digital images during image acquisition, coding, transmission, and processing steps. Filtering image data is a standard process used in almost every image processing system. Here, we use a Gaussian sharpening kernel to reduce noise. Gaussian sharpening is an image sharpening technique that utilizes a blurred, or unsharp negative picture to make a mask of the original image. The unsharp-masking is then joined with the original positive image, making an image that is less blurry than the original.

Clahe Enhancement - Contrast limited adaptive histogram equalization is a modified part of adaptive histogram equalization. In this method, the enhancement function is applied over all neighbourhood pixels and the transformation function is derived. This is differing from AHE because of its contrast limiting. CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighbouring tiles are then combined using bilinear interpolation to remove the artificial boundaries.

In our project, Clahe[15] is used by converting RGB colour image into LAB colour model(L for lightness and a and b for the colour opponents green–red and blue-yellow) and applying Clahe on the Lightness channel.

(ii) **Face detection**

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belongs to a given class. Face-detection algorithms focus on the detection of frontal human faces.

In this project, face detection is performed using the get_frontal_face_detector() function in dlib library. Dlib is an open-source suite of applications and libraries written in C++ that offers a wide range of functionality across a number of machine learning sectors, including classification and regression and numerical algorithms. the get_frontal_face_detector() function uses a HOG + Linear SVM face detector algorithms that is accurate and computationally efficient.

After face detection, facial landmarks are detected using dlib's facial landmark detector. The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face. The indexes of the 68 coordinates can be visualized in the image below:

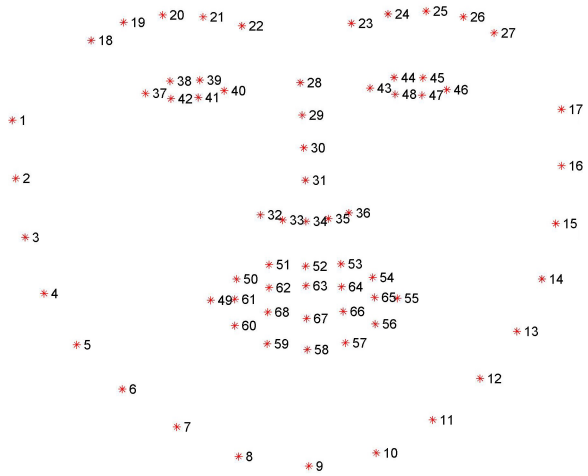


Figure 5.4: The 68 facial landmarks coordinates

(iii) Lip localization

In this study, we use the Dlib software to execute the shape_predictor_68_face_landmarks model to perform facial contour recognition[10]. In shape_predictor_68_face_landmarks, it applied the histogram of oriented gradients (HOG) and Support Vector Machines (SVM) algorithm. The histogram of oriented gradient is an image processing algorithm that performs a feature extraction function. It will have the information of 68 points mark-up in the facial contour, if there is a new input frame it will have the similarity on the face and mark the 68 points. Support Vector Machines (SVM) is a machine learning algorithm. HOG data is used to classify the landmark.

In this study, facial markings from 49 to 67 (the mouth section) are cropped for further process.

5.2.3 Feature Extraction

In this phase, we take the HOG, SURF and HAAR features[5] from the input image. Each feature extraction is done separately.

(i) **The histogram of oriented gradients (HOG) -** It is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.

(ii) **Speeded up robust features (SURF) -** It is a local feature detector and descriptor. It can be used for tasks such as object recognition, image registration, classification, or 3D reconstruction. To detect interest points, SURF uses an integer approximation of the determinant of the Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image. Its feature descriptor is based on the sum of the Haar wavelet response around the point of interest. These can also be computed with the aid of the integral image.

(iii) **Haar features -** A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For example, with a human face, it is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore, a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

5.2.4 Word prediction using GRU-based neural network

A Gated Recurrent Unit is a variant of the Recurrent Neural Network (RNN) design and employs a gated process to control and manage the flow of information between cells in the neural networks. Introduced in 2014 by Cho, et al.[3], GRU facilitates capturing dependencies from huge sequential data without excluding information from the prior portion of the series of data. This is performed by its gated units that solve exploding/vanishing gradient problems of traditional RNNs. Such gates control the information that needs to

be discarded or maintained on each step.

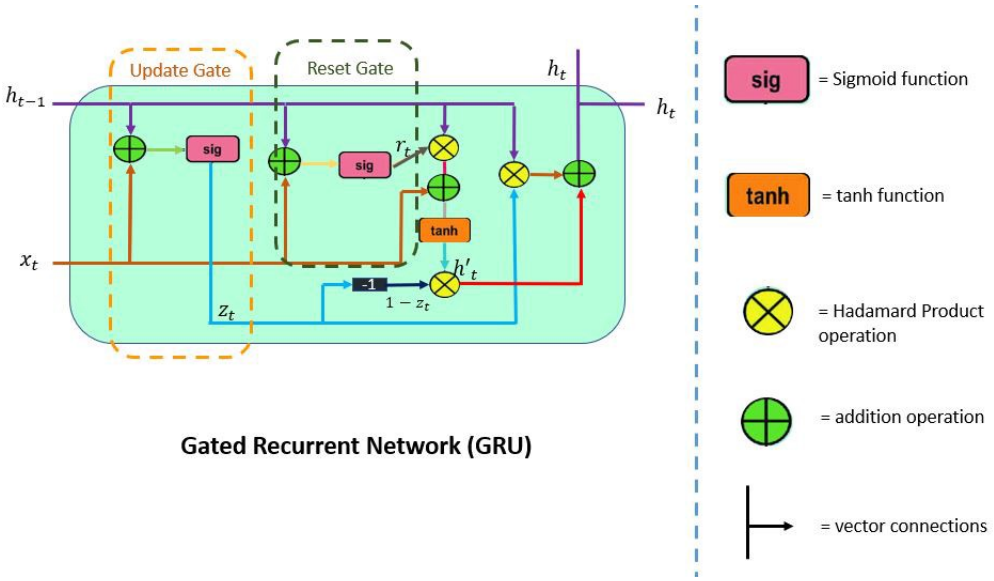


Figure 5.5: Gated Recurrent Network

To solve the vanishing gradient problem of a standard RNN, GRU uses, the so-called, update gate and reset gate[2]. The different gates of a GRU are as described below:-

1. Update Gate(z): It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit. It is computed as $z_t = \sigma(W_z x_t + U_z h_(t-1)$.

2. Reset Gate(r): It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit. It is computed as $r_t = \sigma(W_r x_t + U_r h_(t-1)$.

3. Current Memory Gate(h'_t): It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input zero-mean. It is computed as $h'_t = tahn(W_t + U(r_t \odot h_(t-1)$.

4. Final memory at current time steps(h_t): As the last step, the network needs to calculate h_t — vector which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content h'_t and what from the previous steps h_{t-1}. It is computed as

$$h_t = (1 - z_t)h_(t - 1) + z_t h'_t$$

In our model, we use three separate GRU networks for processing the HOG, SURF and Haar features. Based on features extracted, the neural network predicts the word and a joint consensus is formed between the three units and finally, the exact word is decoded.

5.2.5 Prediction

For prediction, we take input videos, process them into frames and per- form feature extraction. Then we load the saved models and get the predictions from them. The model predicts each word spoken by the speaker.

Chapter 6

SYSTEM REQUIREMENTS

6.1 Hardware

- Processor: i3 or i5
- RAM: 8GB (Minimum)
- Hard Disk: 500GB or above

6.2 Software

- Tool: Python IDLE
- Python: version3
- Operating System: Windows 7 or later
- Libraries:
  - Front End: Tkinter
  - OpenCV
  - Matplotlib

- Tensorflow
- Keras
- dlib

6.3 Brief description of technologies and tools

6.3.1 Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

6.3.2 Anaconda

Anaconda distribution is a free and open-source platform for Python/R programming languages. It can be easily installed on any OS such as Windows, Linux, and MAC OS. It provides more than 1500 Python/R data science packages which are suitable for developing machine learning and deep learning models.

Anaconda distribution provides installation of Python with various IDEs such as Jupyter Notebook, Spyder, Anaconda prompt, etc. Hence it is a very convenient packaged solution that you can easily download and install on your computer. It will automatically install Python and some basic IDEs and libraries with it.

6.3.3 Deep learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the

human brain, allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

6.3.4 Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de-facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from the Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

6.3.5 OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting in 2011, OpenCV features GPU acceleration for real-time operations.OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE.

6.3.6 Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on the training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

6.3.7 Dlib

DLib is an open-source C++ library implementing a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction. The dlib library is arguably one of the most utilized packages for face recognition. A Python package appropriately named face_recognition wraps dlib's face recognition functions into a simple, easy-to-use API.

6.3.8 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. It is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.Keras provides a complete framework to create any type of neural network.

6.4 Use Case Diagram

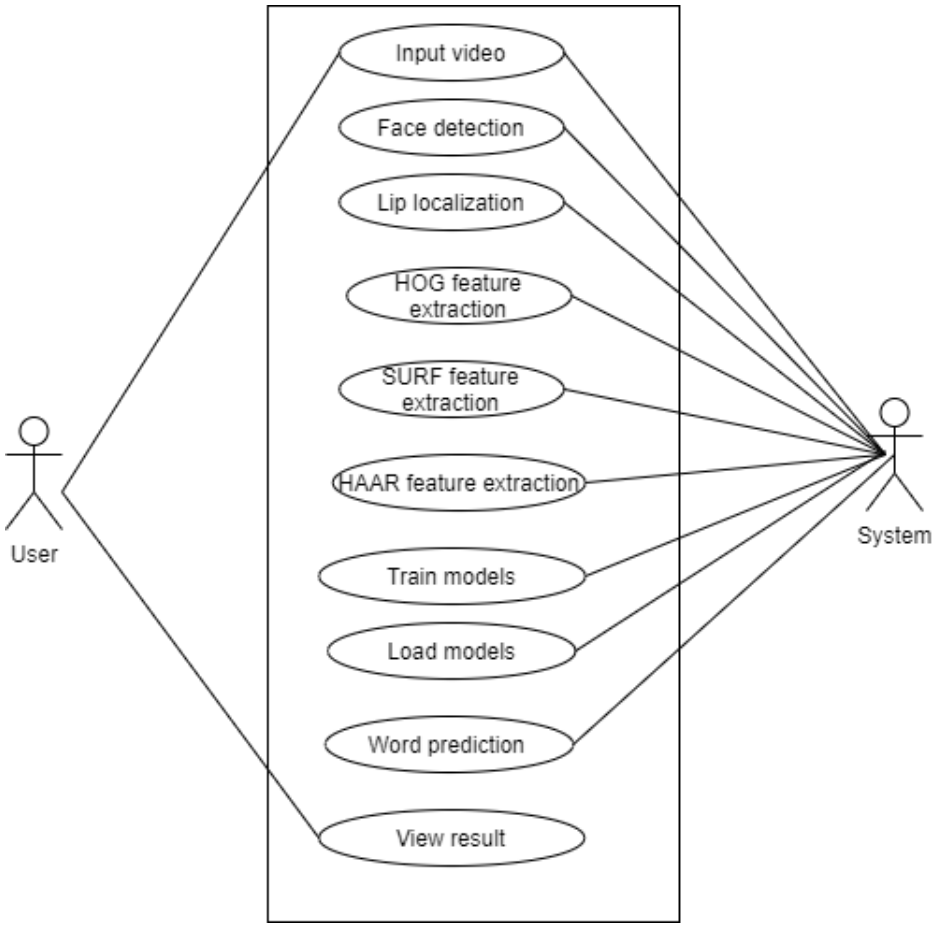


Figure 6.1: Use case diagram

Chapter 7

IMPLEMENTATION

7.1 Data Flow Diagram

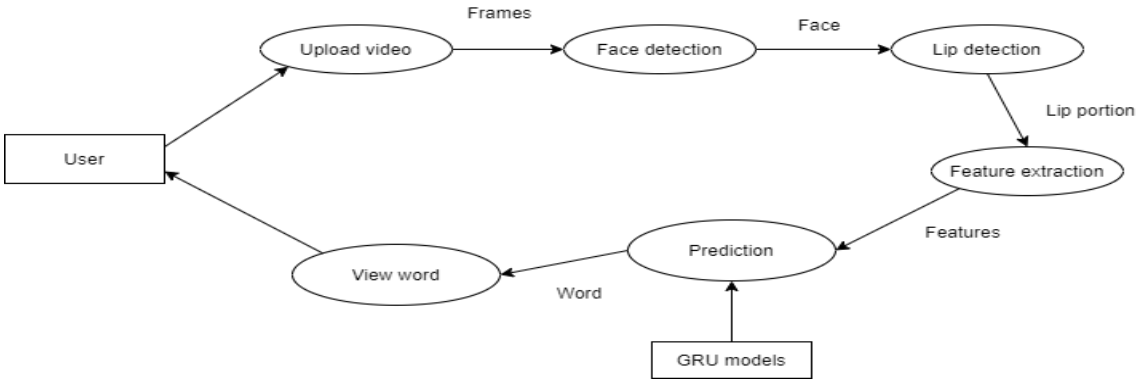


Figure 7.1: DFD of our lip reading system

7.2 Loading MIRACL-VC1 dataset

After importing the necessary python libraries, set the path to the dataset folder. The folder is accessed using the os.scandir() method. Videos having frames below 7 are discarded. The dataset contains the frames of each speaker speaking 10 words 10 times.

```
def read_from_folder(path):
    print("============================================")
    print("Scanning Folders....")
    data1=[]
    data2=[]
    data3=[]
    labels=[]
    subfolders= [f for f in os.scandir(path) if f.is_dir()]
    for sf in subfolders:
        words=[f for f in os.scandir(sf.path+'/'+'words') if f.is_dir()]
        c=0
        for w in words:
            samples=[f for f in os.scandir(w.path) if f.is_dir()]
            for s in samples:
                print(s.path)
                imgs=[f for f in os.scandir(s.path)]
                dl1=[]
                dl2=[]
                dl3=[]
                for j in imgs:
                    print(j.name)
                    if j.name.split('.')[1]=='png':
                        continue
                    path1=path+'/'+sf.name+'/'+'words/'+w.name+'/'+s.name+'/'+j.name
                    # print(path1)
                    d1,d2,d3=process(path1)
                    # print(d1.shape)
                    # print(d2.shape)
                    # print(d3.shape)
                    dl1.append(d1)
                    dl2.append(d2)
                    dl3.append(d3)
                try:
                    dl1=np.array(dl1[:7])
                    dl2=np.array(dl2[:7])
                    dl3=np.array(dl3[:7])
                    assert dl1.shape == (7,40,40)
                    assert dl2.shape == (7,40,40,3)
                    assert dl3.shape == (7,160,160)

                    print(dl1.shape,'=======')
                    data1.append(dl1)


                    #print(dl2.shape,'=======')
                    data2.append(dl2)


                    #print(dl3.shape,'=======')
                    data3.append(dl3)

                    labels.append(c)
                except Exception as e:
                    print(e)
                    print(10*'#')
                    continue
            c+=1

    print('found ',len(data1),' images belonging to',len(subfolders),' classes')
    print("converting to dataframe...")

    return np.array(data1),np.array(data2),np.array(data3),np.array(labels)
```

Figure 7.2: Loading MIRACL-VC1 dataset

7.3 Preprocessing the frames

For pre-processing, we used Gaussian kernel and Clahe enhancement. **process()** function contains the following:

The Gaussian kernel values are stored as numpy array and stored in the **kernel** variable. The **filer2d()** function from OpenCV convolves the image with the kernel. The parameters

are: src - source image to apply the filter on, ddepth - depth of the output image [-1 will give the output image depth as same as the input image], kernel - the 2D matrix we want the image to convolve with.

Now to apply Clahe enhancement, the RGB color image is converted to LAB color code using **lab= cv2.cvtColor(img, cv2.COLOR_BGR2LAB)**. Then it is split into 3 channels as Clahe needs to be applied in the l(lightness) channel. The clahe object is created using **clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))**, where clipLimit - the threshold for contrast change and titleGridSize - sets the number of tiles in rows and columns which is by default 8*8. **clahe.apply() function applies clahe to l channel**. The clahe applied channel (cl) is merged with a and b. After that the image is converted to RGB color code.

```
def process(path):
    img=cv2.imread(path)
    kernel = np.array([[0, -1, 0],
                       [-1, 5,-1],
                       [0, -1, 0]])
    img = cv2.filter2D(src=img, ddepth=-1, kernel=kernel)
    lab= cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
    cl = clahe.apply(l)
    limg = cv2.merge((cl,a,b))
    dst=cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)

    cv2.imwrite('eh.jpg',dst)
    dst,dst1,dst2=crop_mouth(dst)
    return dst,dst1,dst2
```

Figure 7.3: Preprocessing of images

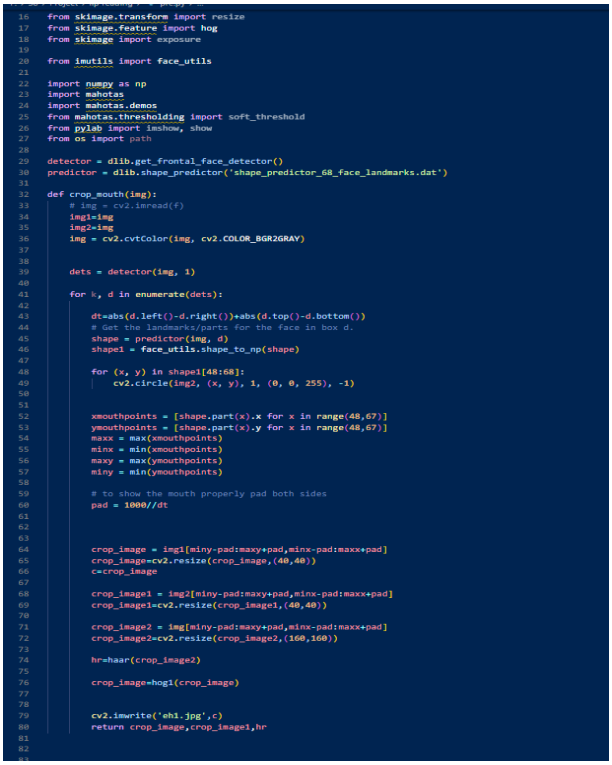
7.4 Face detection and lip region cropping

The **crop_mouth()** function contains the code for face detection and lip region cropping. The images are stored in three separate variables to apply three features in the next step. For face detection **dlib.get_frontal_face_detector()** is used. It returns an array of arrays of rectangle objects that represent a rectangular area of an image.

dlib.shape_predictor() is a tool that takes in an image region containing some object

and outputs a set of point locations that define the pose of the object. Here we use the **shape_predictor_68_face_landmarks.dat** model to create the predictor object. We then pass the frame and the detected rectangle dimensions. Then lip region landmark points are from 48 to 67. The detected landmarks are then plotted on the frame using cv2.circle

To show the mouth properly both sides of the frame are padded. Then lip region is cropped from each frames and stored as three variables after resizing: **crop_image**, **crop_image1**, **crop_image2**. The cropped images are passed to the feature extraction functions and the crop_mouth() function returns the result of feature extraction as numpy array files.



```
from skimage.transform import resize
from skimage.feature import hog
from skimage import exposure

from imutils import face_utils

import numpy as np
import mahotas
import mahotas.demos
from mahotas.thresholding import soft_threshold
from pylab import imshow, show
from os import path

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

def crop_mouth(img):
    # img = cv2.imread(f)
    img1=img
    img2=img
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


    dets = detector(img, 1)

    for k, d in enumerate(dets):

        dt=abs(d.left()-d.right())+abs(d.top()-d.bottom())
        # Get the landmarks/parts for the face in box d.
        shape = predictor(img, d)
        shape1 = face_utils.shape_to_np(shape)

        for (x, y) in shape1[48:68]:
            cv2.circle(img2, (x, y), 1, (0, 0, 255), -1)


        xmouthpoints = [shape.part(x).x for x in range(48,67)]
        ymouthpoints = [shape.part(x).y for x in range(48,67)]
        maxx = max(xmouthpoints)
        minx = min(xmouthpoints)
        maxy = max(ymouthpoints)
        miny = min(ymouthpoints)

        # to show the mouth properly pad both sides
        pad = 1000//dt



        crop_image = img1[miny-pad:maxy+pad,minx-pad:maxx+pad]
        crop_image=cv2.resize(crop_image,(40,40))
        c=crop_image

        crop_image1 = img2[miny-pad:maxy+pad,minx-pad:maxx+pad]
        crop_image1=cv2.resize(crop_image1,(40,40))

        crop_image2 = img[miny-pad:maxy+pad,minx-pad:maxx+pad]
        crop_image2=cv2.resize(crop_image2,(160,160))

        hr=haar(crop_image2)

        crop_image=hog1(crop_image)


        cv2.imwrite('eh1.jpg',c)
        return crop_image,crop_image1,hr
```

Figure 7.4: Face detection and lip localization

7.5 Feature Extraction

hog() function in scikit-image library is used for extracting the hog features. The parameters are: img - input image, orientations- number of orientation bins, pixels_per_cell - size (in pixels) of a cell, cells_per_block- number of cells in each block, multichannel- if True, the last image dimension is considered as a colour channel, otherwise as spatial.

mahotas.haar() function extracts Haar features from the greyscale image. The Haar wavelet is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis. The function returns an image object.

cv2.xfeatures2d.SURF_create(400) creates SURF object using OpenCV library and **surf.detectAndCompute(img,None)** applies the surf function on the image to get the keypoints and descriptors.

```
def hog1(img):
    fd, hog_image = hog(img, orientations=6, pixels_per_cell=(2, 2), cells_per_block=(3, 3), visualize=True, multichannel=True)
    return hog_image

def surf1(img):
    surf = cv2.xfeatures2d.SURF_create(400)
    kp, des = surf.detectAndCompute(img,None)

def haar(img):
    h = mahotas.haar(img)
    return h
```

Figure 7.5: Feature extraction

7.6 GRU model creation and Training

7.6.1 Model creation

GRU models are created for three features separately using the Keras library. Keras provides the **SGD** class that implements the stochastic gradient descent optimizer with a

learning rate and momentum. The default learning rate is 0.01 and no momentum is used by default.

Input() is used to instantiate a Keras tensor. A Keras tensor is a tensor object from the underlying backend, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model. The parameter shape takes values 40*40 pixels for haar and SURF and 160*160 for HOG.

Conv2D() is the 2D Convolution Layer, creating a convolution kernel that is wind with layers input which helps produce a tensor of outputs. filters parameter determines the number of output filters in the convolution.

Flatten() is the first flatten layer applied on the input image. **Model()** groups layers into an object with training and inference features.

Sequential() - The GRU RNN is a Sequential Keras model. So first Sequential model is defined and the GRU layer is added to it. **TimeDistributed()** layer helps in applying the GRU layer to every temporal slice of the input.

GRU() is the Gated Recurrent Unit layer. It creates a 100-cell layer. The parameter return_sequences specifies whether to return the output sequence or not.

Flatten()(model1.output) is the second flatten layer.

Dense() - The first Dense layer has 512 output nodes, uses relu activation function and is applied to the flatten layer output. The second Dense layer has 10 output nodes(as the MIRACL-VC1 dataset has 10 classes of words), uses the softmax function and is applied to the output of the previous dense layer.

model.compile() configures the model for training.

```
from keras.layers.merge import concatenate
from keras.models import Model, Sequential
from keras.layers import Dense, Input,Conv2D,MaxPool2D,GlobalAveragePooling2D,LSTM,Flatten,Input
from keras.layers import TimeDistributed, GRU, Dense, Dropout,Activation,GlobalMaxPool2D,Concatenate,BatchNormalization
from keras.optimizers import SGD

opt = SGD(lr=0.01)
def main2(shape):
    #  with (112, 112, 3) input shape
    img_input = Input(shape=(40,40,3))
    input1=Conv2D(32,3)(img_input)
    input2=Flatten()(input1)

    m1=Model(img_input, input2)

    # then create our final model
    model1 =Sequential()
    # with (5, 112, 112, 3) shape
    model1.add(TimeDistributed(m1, input_shape=shape))

    model1.add(GRU(100,return_sequences=True))

    x0=Flatten()(model1.output)

    x=Dense(512,activation='relu')(x0)
    x1=Dense(10,activation='softmax')(x)


    model = Model(inputs=model1.input, outputs=x1)
    model.compile(optimizer='adam', loss='binary_crossentropy',
                  metrics=['accuracy'])
    model.summary()
    return model
```

Figure 7.6: GRU model creation

7.6.2 Training

To train the model in Keras, the model.fit() function. To use the fit function, we need to pass in the training data for x and y, the validation, the batch_size, and the epochs. The dataset is split as 80% for training and 20% for testing. The model is trained separately on three features for 10 epochs and the model is saved.

Chapter 8

EXPERIMENTAL RESULTS

8.1 Final output

When the program is run the following login page appears.

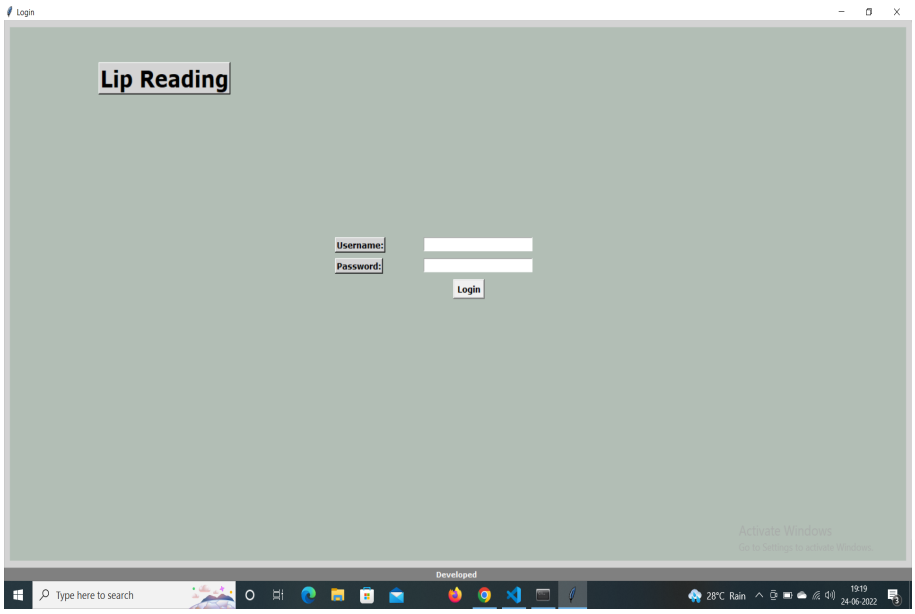
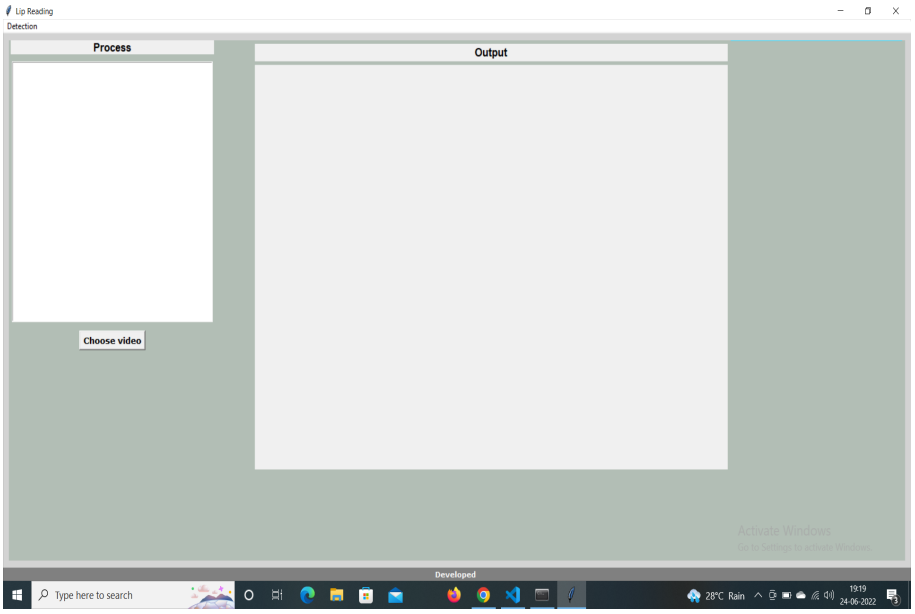


Figure 8.1: Login Page

Entering the username and password brings the main window into action.



The m

Figure 8.2: The main window of Lip reading system

When clicked on choose video an option appears for choosing a video from a list of several test videos.

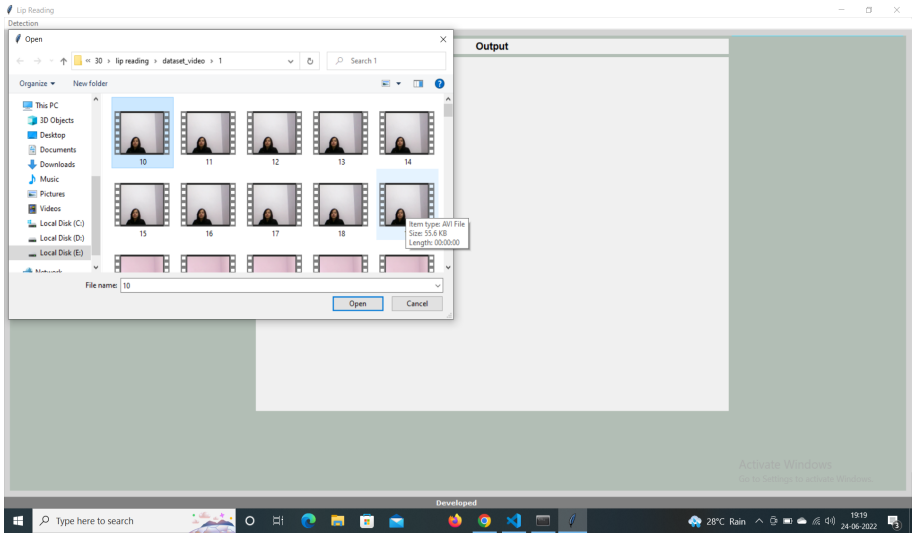


Figure 8.3: Choosing input video

Frames are read one by one and are displayed on the screen.

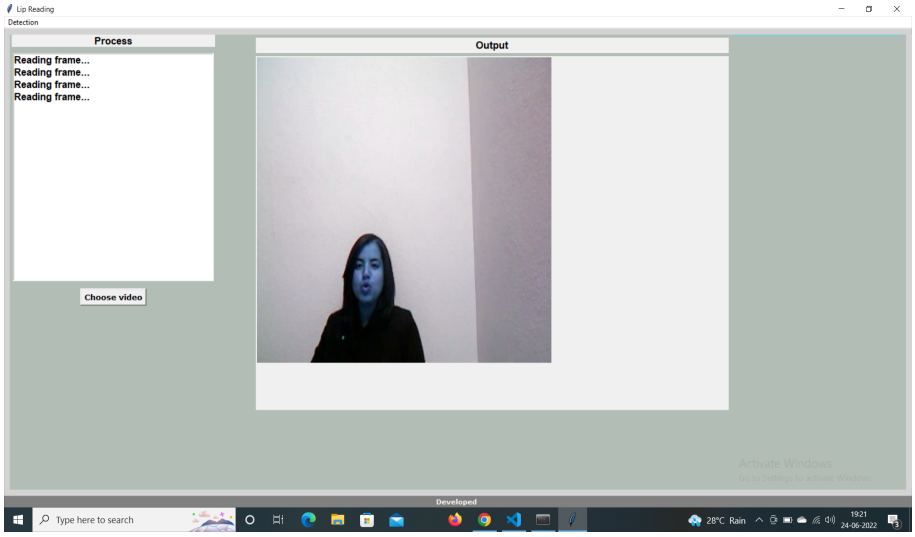


Figure 8.4: Reading frames from the input video

All the frames are read and the output is displayed on the screen

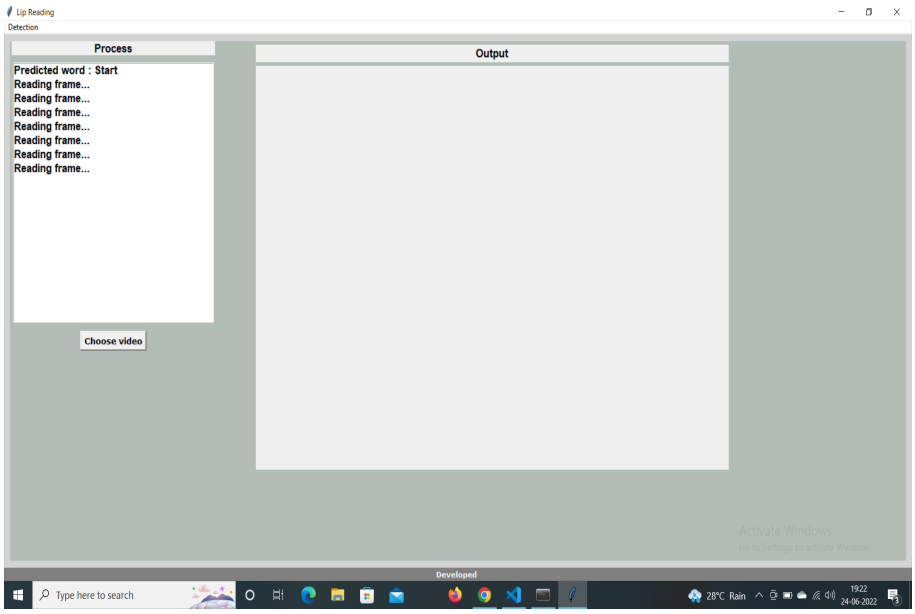


Figure 8.5: Word detected from the input video

8.2 Performance Evaluation

The performance of a classifier is evaluated using a confusion matrix. A confusion matrix is a matrix which shows the actual vs predicted results.

Since our problem is a multi class problem with 10 classes,the confusion matrix formed will have 10 rows and 10 columns.The figure given below shows the confusion matrix.

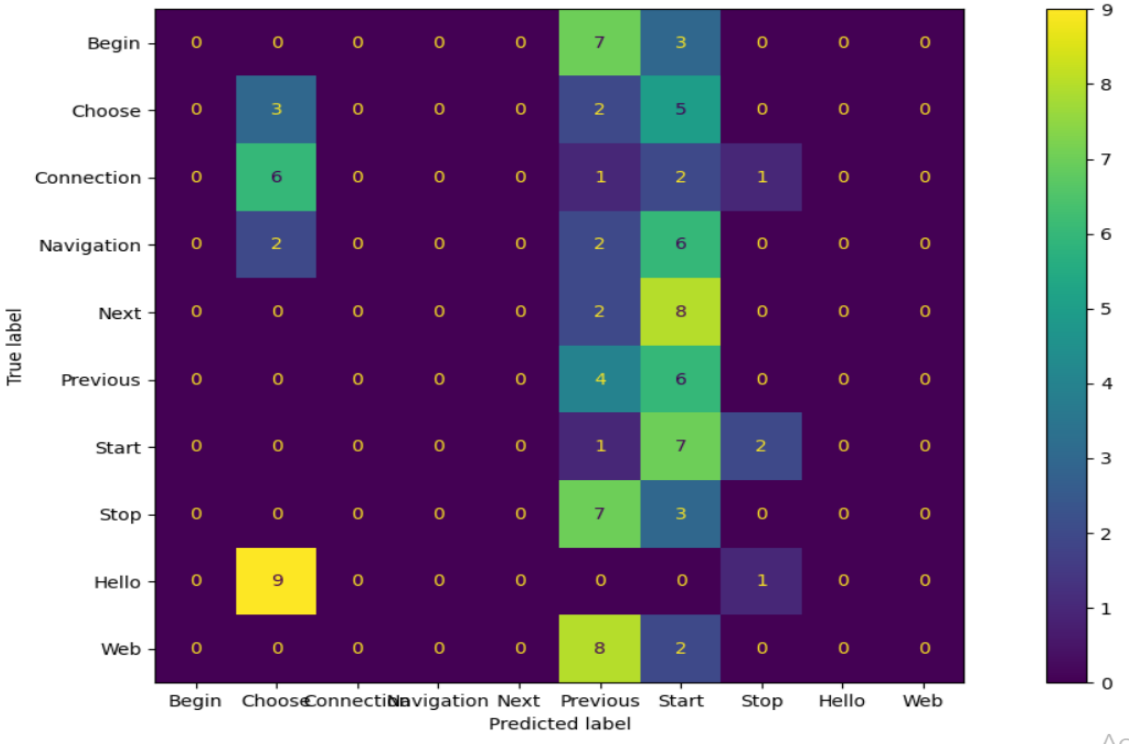


Figure 8.6: Confusion matrix

Chapter 9

CONCLUSION

Lip reading system based on Deep learning is an ongoing research topic and has attracted a lot of attention in recent years. While there has been substantial progress in this field, a lot is yet to be achieved in regards to the accuracy and vocabulary range of lip-reading models. In this project, the prediction of words from the lip movement of the speakers is done using the GRU networks successfully. Here, the prediction is carried out from the series of the image dataset of the lip region. The investigational dataset consists of 10 women and 5 men, a total of 15 speakers.

However, it is an undeniable fact that with the increase in processing potent of computers and the availability of good datasets combined with new techniques the day is not far when we can build a cheap automated lip-reading device that can predict with utmost precision. In further research, the lipreading dataset model is studied on the real-time broadcast shows and videos from news, debates and discussions. The combination of an acoustic and a visual system results in higher accuracy than for each system alone. It would therefore be possible to combine the model designed in this project with an acoustic ASR system to increase the accuracy of both

REFERENCES

[1] Yannis M Assael et al. "Lipnet: End-to-end sentence-level lipreading". In: *arXiv preprint arXiv:1611.01599* (2016).

[2] Linnar Billman and Johan Hullberg. *Speech Reading with Deep Neural Networks*. 2018.

[3] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[4] SM Mazharul Hoque Chowdhury et al. "Text Extraction through Video Lip Reading Using Deep Learning". In: *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*. IEEE. 2019, pp. 240–243.

[5] Mohamed Ezz, Ayman Mohamed Mostafa, and Abdurrahman A Nasr. "A Silent Password Recognition Framework Based on Lip Analysis". In: *IEEE Access* 8 (2020), pp. 55354–55371.

[6] Souheil Fenghour et al. "Deep learning-based automated lip-reading: A survey". In: *IEEE Access* (2021).

[7] Souheil Fenghour et al. "Lip Reading Sentences Using Deep Learning with Only Visual Cues". In: *IEEE Access* 8 (2020), pp. 215516–215530.

[8] Ahmad Basheer Hassanat. "Visual words for automatic lip-reading". In: *arXiv preprint arXiv:1409.6689* (2014).

[9] Brais Martinez et al. "Lipreading using temporal convolutional networks". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6319–6323.

[10] Hea Choon Ngo et al. "A Pipeline to Data Preprocessing for Lipreading and Audio-Visual Speech Recognition". In: *International Journal* 9.4 (2020).

[11] Dário Augusto Borges Oliveira, Andrea Britto Mattos, and Edmilson Da Silva Morais. "Improving Viseme Recognition Using GAN-Based Frontal View Mapping." In: *CVPR Workshops*. 2018, pp. 2148–2155.

[12] M. M. Ramya and F. M. Ahmed. "Detection of Words from Lip-Movement of Speakers Using Deep Learning". In: 2020.

[13] Ahmed Rekik, Achraf Ben-Hamadou, and Walid Mahdi. "A New Visual Speech Recognition Approach for RGB-D Cameras". In: *Image Analysis and Recognition - 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014*. 2014, pp. 21–28.

[14] Kai Xu et al. "LCANet: End-to-end lipreading with cascaded attention-CTC". In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE. 2018, pp. 548–555.

[15] Garima Yadav, Saurabh Maheshwari, and Anjali Agarwal. "Contrast limited adaptive histogram equalization based enhancement for real time video system". In: *2014 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2014, pp. 2392–2397.