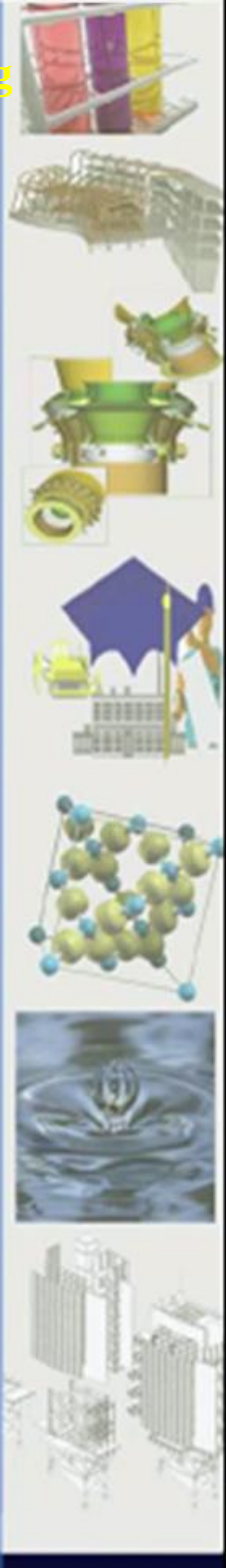
Faculty of Engineering - Cairo University
Credit Hour System Programs

Communication and Computer Engineering

CCE-C

Graduation Project Report
Fall/2020

Lipify



Prepared by:

Ahmed Mohamed Salah
Amr Khaled Zaky
Mostafa Abdelgawad Mohamed
Mustafa Mufeed Abdul Majeed

Supervised by:

Prof. Dr. Nevin Darwish

Cairo University
Faculty of Engineering
Department of Computer Engineering

Lipify

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

In Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Ahmed Mohamed Salah
Amr Khaled Zaky
Mostafa Abdelgawad Mohamed
Mustafa Mufeed Abdul Majeed

Supervised by

Prof. Dr. Nevin Darwish

Thursday, 9 July, 2020

Project Code	**GP-[CCE-C]-[2020]**	
Project Title	**Lipify**	
Keywords	Machine Learning, Artificial Intelligence, Image Processing, Mobile Application	
Students	**Name:** Ahmed Mohamed Salah **Email:** ahmed.mohamed.salah@outlook.com **Phone:** +20 114 980 0442 **Address:** 6th of October City 	**Name:** Amr Khaled Zaky **Email:** amrkh97@gmail.com **Phone:** +20 114 183 7032 **Address:** El-Maadi
	Name: Mostafa Abdelgawad Mohamed **Email:** elgawad.mostafa@gmail.com **Phone:** +20 112 921 6108 **Address:** Haram 	**Name:** Mustafa Mufeed AbdulMajeed **Email:** m.mufeed1996@gmail.com **Phone:** +20 109 414 4355 **Address:** Dokki
Supervisor	Prof. Dr. Nevin Darwish	**Email:** ndarwish@ieee.org
	Signature:	**Phone:** +20 122 224 7364
Project Summary	The project aims to help the hearing-impaired communicate better with their surroundings. Our main goal is to allow them to grasp what is uttered through reading lips movement only. It consists of lips extraction module, learning module, and a cross-platform mobile application interface. We currently support English language only with certain categories, but we are looking to expand our language support as we go.	

Abstract

Since the beginning of time, communication between living things has been necessary to understand one another. Communication can be through talking, hand signs and/or light signals. The hearing-impaired have difficulties while communicating with those who do not know hand signs. Therefore, and due to technological advances in our era, we are aiming to make it easy for both sides. For the hearing-impaired the project can be used in two simple steps. In the first step, a user may imitate lip movement of words and for the next step the user is able to grasp the words uttered.

The project has been successfully implemented and tested taking advantage of Machine Learning, Computer Vision, and the wide availability and usability of mobile phones to predict what a person wants to say by tracing their lip movement. To be specific, it consists of lips extraction module, learning module, and a cross-platform mobile application interface however the current version only supports a limited, number of English language words with specific categories.

الملخص

تعتمد جميع المخلوقات الحية منذ فجر التاريخ على طرق عدة للتواصل لكي تعيش هذه المخلوقات مع بعضها البعض. فالإنسان مثلا له طرق عدة للتواصل مثل الكلام بلغات عدة ولغة الاشارة والضوء. وبما أن الكلام يصدر عنه صوت ولأن لغة الاشارة لا يتقنها كثير من الناس، يعاني أصحاب السمع الضعيف في التواصل مع من حولهم من الناس.

وبهذا الهدف نصب أعيننا، قررنا استغلال التطور التكنولوجي الذي يتميز به عصرنا هذا لصنع برنامج للهواتف الذكية المحمولة للتسهيل على ضعاف السمع التواصل مع من حولهم باستخدام الذكاء الاصطناعي وطرق معالجة الصور وتطوير التطبيقات. وقد تم تنفيذ المقترح واختباره بنجاح فقد تم بناء نظام يتسم فى اصداره الاولى هذا بالتعامل مع عدد محدود من الكلمات باللغة الانجليزية.

يلخص هذا التقرير تجربتنا فى بناء النظام حيث تقدم خط سيرنا في صناعة هذا المشروع وطرق تنفيذ كل وحدة من وحداته مثل وحدة الذكاء الاصطناعي ووحدة معالجة الصور ووحدة واجهة المستخدم. كما يحتوى أيضا على المشاكل التي واجهتنا وكيف تخطيناها. بحيث نمكن القارئ من بناء نظام مماثل اذا رغب فى ذلك.

يتكون هذا التقرير من عدة فصول لكل فصل منهم موضوعه الخاص. فيتناول الفصل الأول المقدمة ولماذا وقع إختيارنا على هذا المشروع بالتحديد وماهي النتائج المرجوة منه وماهي المشكلة التي نحاول حلها. ثم ننتقل الى الفصل الثاني الذي يشرح من المستهدف بالضبط من هذا المشروع وينتقل بعد ذلك الى دراسة السوق والتحليل المالي. ثم يغوص بنا الفصل الثالث في الدراسة الأدبية التي تكمن في خلفيات عدة من أبعاد المشروع مثل خلفية عن تعليم الآلة واستخراج المميزات ومعالجة الصور وتطوير برامج الهواتف المحمولة وماهي الآليات التي نفذناها.

ثم يتطرق الفصل الرابع الى دراسة الجدوى ومن ثم تحضير مجموعة البيانات في الفصل الخامس وكيف جعلناها مستعدة لتعديلاتنا. ثم نصف بالتفصيل اسلوب بناء النظام المقترح في الفصل السادس بالاضافة الى وصف كل وحدة بالتفصيل ومن هذه الوحدات: وحدة معالجة الصور ووحدة التعلم ووحدة واجهة الجوال المحمول. وفي الوحدة السابعة نتطرق الى اسلوب البناء الذي نفذناه. وفي الفصل الثامن نعرض واجهتي النظام المنفذ للهواتف المحمولة والانترنت. واختبارات النظام المتكامل مذكورة في الفصل التاسع ونختتم البحث في الفصل العاشر بالخلاصة والمقترحات البحثية والتطبيقية المستقبلية.

Acknowledgement

First and foremost, we would like to thank our supervisor Prof. Dr. Nevin Darwish for her encouragement, support, and guidance throughout this project. In addition to that, we want to thank the teaching staff of the Faculty of Engineering, Cairo University, who gave us the needed knowledge and information both within and outside the courses. We learned so much from them on both academic and personal levels.

Very special thanks to our families, friends, and loved ones, who have tolerated us in difficult times and shared the joy in the good ones. Your support and encouragement made this journey fun and bearable to all of us. We dedicate this project to all people having any kind of hearing-impair across the globe wishing them a speedy recovery.

Ahmed, Amr, Mustafa, Mostafa

Table of Contents

Abstract iii
الملخص iv
Acknowledgement v
Table of Contents vi
List of Figures x
List of Tables xii
List of Abbreviations xiii
Contacts xiv
Chapter 1 : Introduction 1
1.1. Motivation and Justification 1
1.2. The Essential Question 1
1.3. Project Objectives and Problem Definition 1
1.4. Project Outcomes 2
1.5. Document Organization 2
Chapter 2 : Market Visibility Study 3
2.1. Targeted Customers 3
2.2. Market Survey 3
2.3. Business Case and Financial Analysis 3
2.3.1. Business Case 3
2.3.2. Financial Analysis 4
2.4. Feasibility Study 5
2.4.1. Description of Products and Services 5
2.4.2. Technology Consideration 5
2.4.3. Products/Services Marketplace 5
2.4.4. Schedule 6
2.4.5. Findings and Recommendations 6
Chapter 3 : Literature Survey 7
3.1. Background on Machine Learning 7
3.1.1. Artificial Neural Networks (ANN) 7
3.1.2. Recurrent Neural Network (RNN) 8
3.1.3. Convolutional Neural Networks (CNN) 9
3.1.4. Adam Optimizer 9
3.2. Background on Feature Extraction 10
3.2.1. Co-articulation Phenomenon 10
3.2.2. Visual Features 10
3.3. Background on Image Processing 11

3.3.1. Canny Edge Detector 11
3.3.2. Color Space Functions 11
3.3.3. Morphological Operations....12
3.3.4. Skin Detection Algorithm....13
3.3.5. Dlib 13
3.3.6. Haar Cascade Classifier....13
3.4. Background on Mobile Application Development 14
3.4.1. Hybrid Mobile Development 14
3.4.2. Native Mobile Development....14
3.4.3. Our Mobile Requirements and Choices 15
3.5. Comparative Study of Image Processing....16
3.5.1. Face Detection Using Skin Tone Segmentation 16
3.5.2. Human Face Detection Using Skin Color Segmentation and Watershed Algorithm....17
3.6. Comparative study on Feature Extraction 18
3.7. Implemented Approaches for Image Processing 20
3.7.1. YCbCr/HSV/RGB skin Model Approach: 20
3.7.2. RGB/Normalized RGB skin Model Approach: 20
Chapter 4 : Dataset Preparation and Preprocessing....21
4.1. Data Visualization 21
4.2. General Block Diagram of the System....22
4.3. Data Preparation 22
4.3.1. Capturing Lips 22
4.4. Features Extraction 23
4.5. Video Capturing....23
4.6. Word Processing....24
4.7. Dataset Preprocessing 25
4.8. Dataset Concatenation....27
Chapter 5 : System Design and Architecture....28
5.1. Overviews and Assumptions....28
5.2. System Architecture Block Diagram....28
5.3. Feature Extraction Module 29
5.4. Image Processing Module....29
5.4.1. First Implementation: Dlib....29
5.4.2. Second Implementation: Skin Segmentation for Lips Detection....30
5.4.3. Third Implementation: RGB Skin Segmentation for Lips Detection....32
5.4.4. Fourth Implementation: Skin Segmentation for Lips Detection Implemented Functions 33
5.4.5. Fifth Implementation: Haar Cascade Classifier 39

5.4.6. Sixth Implementation: Hybrid Module 40
5.5. Learning Module 41
5.5.1. Deep Learning using CNNs and RNNs 41
5.5.2. Artificial Neural Networks 42
5.6. Mobile Interface and Server Module 43
5.6.1. Mobile Interface 43
5.6.2. Server 43
Chapter 6 : Implemented Architecture 44
6.1. Adverb CNN Architecture 44
6.1.1. Using VGG-19 44
6.1.2. Building CNN by using Numpy only 45
6.1.3. Using our custom architecture 45
6.2. Models' Architecture 46
6.3. Colors CNN 46
6.3.1. Commands CNN 47
6.3.2. Numbers CNN 48
6.3.3. Prepositions CNN 48
6.3.4. Alphabet CNN 49
Chapter 7 : System Testing and Verification 50
7.1. Testing Setup 50
7.2. Testing Plan and Strategy 50
7.2.1. Module Testing 50
7.3. Testing Schedule 53
7.3.1. Unit Testing Schedule: 53
7.3.2. System Integration Testing Schedule 57
7.3.3. System Testing Schedule 59
Chapter 8 : Conclusion and Future Work 61
8.1. Faced Challenges 61
8.1.1. Dataset Problems: 61
8.1.2. Model Selection: 62
8.2. Gained Experience 62
8.2.1. Scientific Research 62
8.2.2. Software development 62
8.3. Conclusions 63
8.4. Future Work 64
References 65
Appendix A : Development Platforms and Tools 67
A.1. Hardware 67

A.1.1. PC/Laptop 67
A.1.2. Mobile Devices 67
A.2. Software 67
A.2.1. PyCharm 67
A.2.2. Android Studio 67
A.2.3. Visual Studio Code 68
A.2.4. Anaconda 68
A.2.5. Notepad++ 68
Appendix B : User Guide 69
B.1. Using Lipify: The Mobile Version 71
B.2. Using Lipify: The Website Version 78

List of Figures

Figure 2.1 Expenses Table 4
Figure 3.1 Artificial Neural Network 7
Figure 3.2 Recurrent Neural Network 8
Figure 3.3 Convolutional Neural Networks 9
Figure 3.4 Erosion operation 12
Figure 3.5 Dilation operation 12
Figure 3.6 The schematic of the hybrid feature extraction process. 18
Figure 4.1 Sample of Data Set 21
Figure 4.2 General System Block Diagram 22
Figure 4.3 Detected Lips 22
Figure 4.4 Data Preparation 23
Figure 4.5 Video Capturing Steps 24
Figure 4.6 Align file example 25
Figure 4.7 Dataset structure after segmentation 25
Figure 4.8 Sentence uttering cases 26
Figure 4.9 Inaccurate segmentation using Haar Cascade 27
Figure 4.10 Accurate segmentation using Haar Cascade 27
Figure 5.1 System Architecture Block Diagram 28
Figure 5.2 Dlib 68 face landmark points 29
Figure 5.3 Detecting Lips 30
Figure 5.4 RGB model output 32
Figure 5.5 Canny Library vs Canny Coded 33
Figure 5.6 RGB to HSV 34
Figure 5.7 Opening operation comparison 35
Figure 5.8 Sobel filter operators 35
Figure 5.9 Sobel filter gradient output 36
Figure 5.10 Canny modified output 36
Figure 5.11 Comparison of Gaussian filter 37
Figure 5.12 Handmade Module Output 38
Figure 5.13 Haar Features Used for Face Detection 39
Figure 5.14 Mouth and face detected 39
Figure 5.15 Hybrid module failure output 40
Figure 5.16 LipNet’s Proposed Architecture 41
Figure 5.17 Multi-Class Classification Neural Network 42
Figure 6.1 VGG-19 Architecture 44
Figure 6.2 Adverb CNN Architecture 45

Figure 6.3 Colors CNN Architecture 46
Figure 6.4 Color CNN Confusion Matrix 47
Figure 6.5 Commands CNN Confusion Matrix 47
Figure 6.6 Numbers CNN Confusion Matrix 48
Figure 6.7 Prepositions CNN Confusion Matrix 48
Figure 6.8 Alphabet CNN Confusion Matrix 49
Figure 6.9 English Phonemic Chart 49
Figure 7.1 Adverb CNN's Test 50
Figure 7.2 Project Prototype Test 51
Figure 7.3 Project Prototype No Videos Passed 51
Figure 7.4 Modules Average Accuracy on Test Videos 52
Figure 7.5 Modules Average Timing on Test Video 52
Figure 8.1 LRW Dataset Example 61
Figure 8.2 Grid Corpus Example 61
Figure 8.3 Models Accuracy Bar Graph 63
Figure B.1 Lipify in app drawer 69
Figure B.2 Home Screen 70
Figure B.3 Sentence Structure Screen 71
Figure B.4 Help Dialogue 71
Figure B.5 No Categories Selected Popup 72
Figure B.6 Selecting categories 73
Figure B.7 Camera Screen 74
Figure B.8 Loading Screen 75
Figure B.9 Prediction Result 76
Figure B.10 Server Error 77
Figure B.11 Website Homepage 78
Figure B.12 Sentence Structure Page 78
Figure B.13 Sentence Structure Help Dialog 79
Figure B.14 Selecting Categories 79
Figure B.15 Camera Page 80
Figure B.16 Camera Page: Recording Video 80
Figure B.17 Camera Page: Video Preview 81
Figure B.18 Prediction Page: Loading 81
Figure B.19 Server Sent Prediction 82
Figure B.20 Prediction Page: Error 82
Figure B.21 Website viewed on a mobile 83

List of Tables

Table 4.1 Comparison between facial extraction techniques 27

List of Abbreviations

AAM	Active Appearance Model
ANN	Artificial Neural Network
ASL	American Sign Language
ASM	Active Shape Model
BSL	British Sign Language
BPNN	Back Propagation Neural Network
CNN	Convolutional Neural Networks
CSO	Central Statistics Office
DWT	Discrete Wavelet transformation
DBN	Dynamic Bayesian Network
FPS	Frames per Second
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HCIA	Hearing Care Industry Association
HMM	Hidden Markov Model
IDE	Integrated Development Environment
LDA	Linear Discriminant Analysis
NIDCD	National Institution on Deafness and Other Communication Disorders
OS	Operating System
RNN	Recurrent Neural Network (RNN)
SOTA	State of The Art
UI	User Interface
UX	User Experience

Contacts

Team Members

	Email	Mobile Number
Ahmed Mohamed Salah	ahmed.mohamed.salah@outlook.com	+20 114 980 0442
Amr Khaled Zaky	amrkh97@gmail.com	+20 114 183 7032
Mostafa Abdelgawad	elgawad.mostafa@gmail.com	+20 112 921 6108
Mustafa Mufeed	m.mufeed@gmail.com	+20 109 414 4355

Supervisor

	Email	Mobile Number
Prof. Dr. Nevin Darwish	ndarwish@ieee.org	+20 122 224 7364

This page is left intentionally empty

Chapter 1: Introduction

Lipify is a lip-reading mobile application that aims to help the hearing-impaired communicate and interact better with their surroundings. It consists of a lip-extracting module-using image processing, a learning module using machine learning, and a mobile application interface using Google's Flutter. The document is organized into many chapters that cover: ***A brief introduction, background, literature survey, feasibility study, data analysis and preprocessing, GUI and communication.***

1.1. Motivation and Justification

We are encouraged to work on this project as it has the potential to help over ***ten million*** English-speaking deaf and hearing-impaired people all over the world [1]. Based on our research, currently there is no product that could help solve the problem of lip reading at a reasonable cost, there is currently a solution being developed and is planned for release within the next year by a company called Liopa [2] in the UK. Moreover, Samsung has lately released a mobile application that translates Morse code through tapping on screen into automated human sound to make it easy for communication.

Seeing the lack of research in the field motivated us to explore how we can utilize our knowledge to implement a solution that could contribute to aiding the hearing-impaired. The project is also relevant to our interests as it mainly involves two fields of computer engineering like Computer Vision, Neural Networks and Machine Learning in general and Neural Networks in specific.

1.2. The Essential Question

The essential question with relevance to the Vision and Mission of the Faculty of Engineering at Cairo University would be: How can we, using our engineering education, help the hearing-impaired nationally, regionally and internationally to better serve individuals, society and the environment?

1.3. Project Objectives and Problem Definition

The objectives of Lipify is to take advantage of the current technological advance to make the lives of the hearing-impaired better. It also aims at facilitating the everyday activities these people perform with their loved ones. One of the major problems facing the hearing-impaired is communication with people who do not know sign language and may misunderstand them. Sign language is far from universal and leads to lots of misunderstandings; it is not a universal dialect.

Different countries have their own sign language standards, and even the differences between British Sign Language and American Sign Language are significant [3]. Moreover, regional areas have their own specific variations, much like accents in spoken languages, and you get many common misunderstandings. Perhaps the most obvious difference is that ASL uses a One- handed fingerspelling alphabet while BSL uses a two-handed alphabet.

Even professional interpreters can easily sign the wrong word, due to slight changes in meaning or by accidentally using incorrect hand movements. This can be especially problematic in formal or official situations, where mistranslations can cause lasting harm, such as in legal situations or hospital visits. Therefore, our aim to build a smart and intelligent mobile application that is able to see the user's lips and transform their words into text.

1.4. Project Outcomes

The outcome of this project is a software developed mobile application that works on the two most popular mobile device operating systems: Android and iOS. There are no hardware outcomes to this project.

1.5. Document Organization

Chapter 1 and Chapter 2 provide a brief introduction to our work, motivation, outcomes, market and feasibility study moving forward to a necessary literature survey containing background information for the problem in order to lead the reader straightly to the point in Chapter 3. The feasibility study, the system specifications, data visualizations, and preprocessing are then introduced through Chapter 4.

Chapter 5 proceeds to show our system architecture and block diagram. It is then followed by Chapter 6, which discusses our neural networks' architectures and findings alongside confusion matrices for each architecture.

The document illustrates our successful testing steps and procedures in Chapter 7. Then the document is finalized in Chapter 8 with a conclusion and for possible future work.

Chapter 2: Market Visibility Study

In this chapter, we show our results surveying the related market to our project. Other similar tools/platforms are discussed. The chapter discusses our targeted customers, market survey, business case and financial analysis, financial analysis as well as a detailed feasibility study.

2.1. Targeted Customers

As previously mentioned, our targeted customers are the hearing-impaired. Lipify will benefit them by facilitating communication with fellow human beings. In 2019, HCIA [4] estimated there are about 3.6 million Australians with a hearing impairment. Canadian Association of the Deaf [5] estimated there are 3.21 million Canadians with a hard of hearing. CSO [6] have estimated there are 103,676 citizens with hearing difficulties in Ireland. According to the department of health in UK [7] there are 212900 UK citizens with a hard of hearing in 2010. Finally, NIDCD [8] estimated there are approximately 3.75 million Americans with troubles in hearing. By adding all these statistics of the top populated English-speaking countries, we could estimate that there are more than 10.9 million English-speaking hearing-impaired in the world. Version 1 targets them and future versions will be extended to cover Arabic language speakers.

2.2. Market Survey

Since Lipify is a mobile application targeting both Android and iOS, we surveyed the respective application stores of both operating systems, Play Store and App Store. We found no lip-reading application on both stores making Lipify the first lip reading mobile application.

2.3. Business Case and Financial Analysis

We start by illustrating the business case and conclude with the detailed feasibility study.

2.3.1. Business Case

Based on Market survey above since Lipify is the first product of its kind, we think Lipify will sell well over the next 5 years. Setting our price to counter the competition.

2.3.2. Financial Analysis

a) The Capex (Capital Expenditure): We need to pay for powerful computers for development if the demand for the application and its maintenance arise. We also need a strong server for handling increasing users. Currently, this is in the order of 43223 with renting a minimal server needs, however hardware is the only declining price product. We also expect in the near future customers may use the cloud at a reasonable rent as are land phones today.

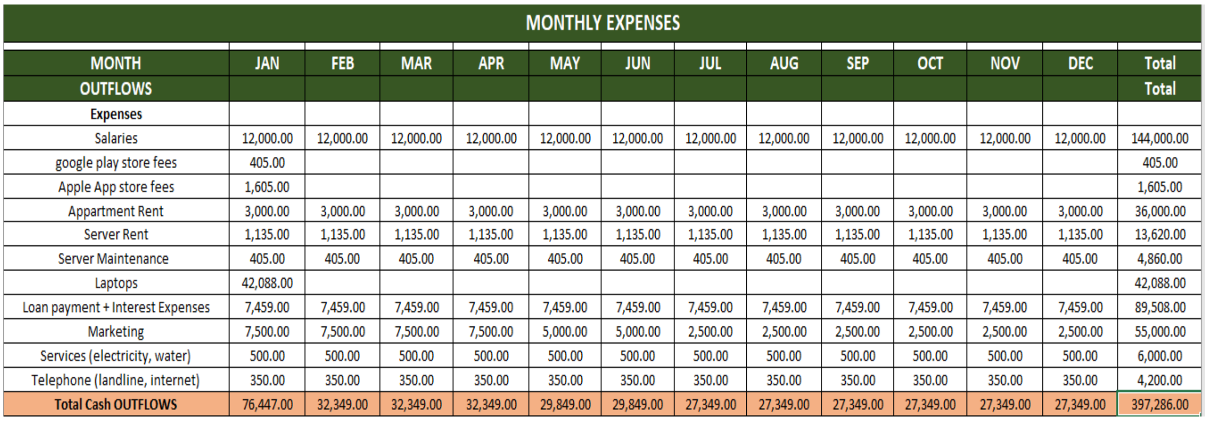
Renting one server will cost around 70 dollars per month; the specifications are minimal server needs. According to servermania.com [9], renting a server for a small business will cost around $70/month where its Specs are: **Intel E3 1240 V3** 32GB RAM, 500GB SSD, Setup in 8 Hours. It is also estimated that server maintenance will cost around 25 dollars per month.

Buying four laptops designed for programming will cost 42088 LE, four programming laptops with around 650 dollars each.

b) The Opex (Operational Expenses): We need to pay salaries if more developers are needed. Also, at the launch of the project, some money will be spent on marketing the application on billboards, TV and radio channels, and social media.

We need salaries for 20 hours of computer engineer developers per day, i.e. 4 engineers working for 5 hours and 25 days a week, to cover necessary version extensions at an estimate of 12000 LE, the average salary for a software engineer in Egypt is 24 LE per hour. At the launch of the project, some money will be spent on marketing the application on billboards, TV and radio channels, and social media. This will add at least 10% more to the cost in the order of 7500 LE.

c) Expenses Requirements



MONTHLY EXPENSES

MONTH	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	Total
OUTFLOWS													Total
Expenses													
Salaries	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	12,000.00	144,000.00
google play store fees	405.00												405.00
Apple App store fees	1,605.00												1,605.00
Appartment Rent	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	3,000.00	36,000.00
Server Rent	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	1,135.00	13,620.00
Server Maintenance	405.00	405.00	405.00	405.00	405.00	405.00	405.00	405.00	405.00	405.00	405.00	405.00	4,860.00
Laptops	42,088.00												42,088.00
Loan payment + Interest Expenses	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	7,459.00	89,508.00
Marketing	7,500.00	7,500.00	7,500.00	7,500.00	5,000.00	5,000.00	2,500.00	2,500.00	2,500.00	2,500.00	2,500.00	2,500.00	55,000.00
Services (electricity, water)	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	6,000.00
Telephone (landline, internet)	350.00	350.00	350.00	350.00	350.00	350.00	350.00	350.00	350.00	350.00	350.00	350.00	4,200.00
Total Cash OUTFLOWS	76,447.00	32,349.00	32,349.00	32,349.00	29,849.00	29,849.00	27,349.00	27,349.00	27,349.00	27,349.00	27,349.00	27,349.00	397,286.00

Figure 2.1 Expenses Table

2.4. Feasibility Study

We are now ready to detail a feasibility study of our project. As previously mentioned, the project is a lip-reading app, in which we capture mouth movement of users, allowing the project to be put in a discussion for further development and production serving many hearing-impaired people for better communication.

2.4.1. Description of Products and Services

The chief goal is to build a system that captures human face, detects the mouth position and traces its movements regarding lips positions and movements in order to predict the words said or willing to be said by the user.

Mobile camera captures the user's video, passing it to a back-end where all the lips tracing will be processed by an already trained module to detect the desired word or group of words. The actual text then propagates to the user's application interface.

Many new initiatives and users can rely on this project. Hearing-impaired users can rely on this application to communicate with those who cannot understand sign language. Moreover, some organizations can use this functionality in order to detect words out of videos with no sound like football games.

2.4.2. Technology Consideration

Considering the technological issues that might happen during the live-phase of the prototype testing and subsequently commercial operation, problems may appear with the mobile camera quality regarding resolution, night sight mode, and similar environmental conditions. Therefore, it might be prudent to operate on devices with medium to high camera capabilities.

Another issue that might arise is the internet speed as well as back-end server speed; since the internet speed in some countries is limited below, certain boundaries thus the application version operated in low-speed internet countries will have the most restricted video time.

2.4.3. Products/Services Marketplace

Our main target will be the largest application stores: Play Store and App Store, using a cross-platform open-source UI software development kit "Flutter". Main users are the ones with issues communicating with those who are unfamiliar with sign language.

One of the recent systems with much similarity to our lip-reading application is a Morse code application introduced by Samsung in 2019, where blind people could use their mobile phones with Morse code taping instead of ordinary texting.

2.4.4. Schedule

The project is expected to take six months from project approval to launch the agreed upon platform. Many of the foundations for this platform are still being discussed for further development. The following is a high-level schedule of some significant milestones for this initiative in the next three months (second semester):

During first 3 weeks, it is expected to deliver a prototype of the application interface (GUI) with an image-processing module to enhance the input image or frame of images to the next module of the system. After week 5, a back-end with a dummy data connecting to the previously introduced GUI is expected to be delivered; making a completely connected application prototype.

After a one-week gap due to midterms. In weeks, seven and eight a complete image-processing module, which extracts the mouth edge points out of the image is expected to be delivered in parallel with a research summary of feature extraction regarding out problem. In weeks, 9 and 10 it is expected to deliver a feature extraction module and a prototype for the machine-learning module with its research summary. Finally, in weeks, 11 and 12 it is expected to assemble all the modules together, solve any new problems and deliver our project-finalized thesis.

2.4.5. Findings and Recommendations

The market survey recommends that the project is currently in demand and profitable. Keeping up with the current technologies and automation, this project will facilitate the communication between people as well as other field's requirements like detecting any unethical words used in a sports event.

Chapter 3: Literature Survey

In this chapter, we give some necessary background that helped us in building ***Lipify***. These backgrounds include facts, theory, algorithms and techniques. Some of the topics discussed are: ***machine learning, feature extraction, image processing, and mobile application development.***

3.1. Background on Machine Learning

Machine learning techniques are quite diverse and could be classified into supervised, unsupervised, semi-supervised and/or reinforcement techniques. We only cover specific supervised techniques that are of interest for our project.

3.1.1. Artificial Neural Networks (ANN)

They are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems, which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as hidden layers consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize [10].

While neural networks (also called "perceptron") have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called "backpropagation," which allows networks to adjust their hidden layers of neurons in situations where the outcome does not match what the creator is hoping for — like a network designed to recognize dogs, which misidentifies a cat, for example. Figure 3.1 shows an example of an Artificial Neural Network showing its input, hidden, and output layers where the input is selected based on the number of input features while the output is based on the number of required classes to classify.

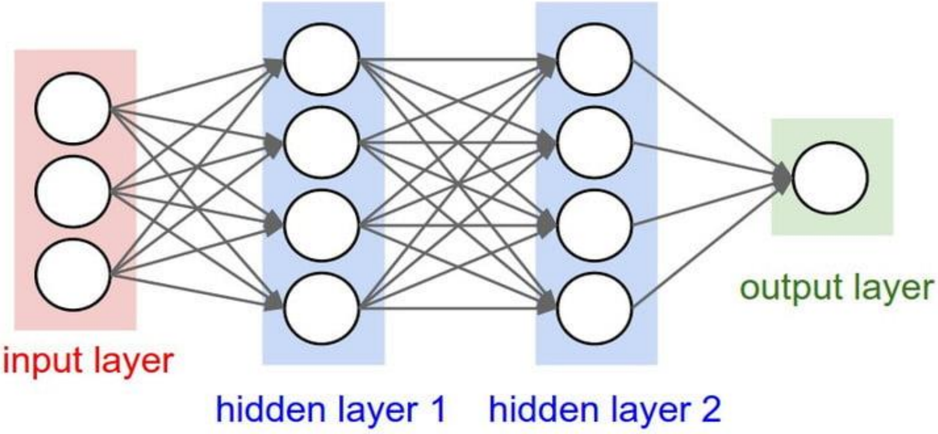


Figure 3.1 Artificial Neural Network

3.1.2. Recurrent Neural Network (RNN)

Humans do not start their thinking from scratch every second. As you read this report, you understand each word based on your understanding of previous words. You do not throw everything away and start thinking from scratch again. Your thoughts have persistence. Traditional artificial neural networks cannot do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It is unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue by allowing for a feedback between layers. They are networks that allow information to persist, where the output from previous step are fed as input to the current step [11]. Figure 3.2 shows a Recurrent Neural Network and how it's 'Recurrent' nature.

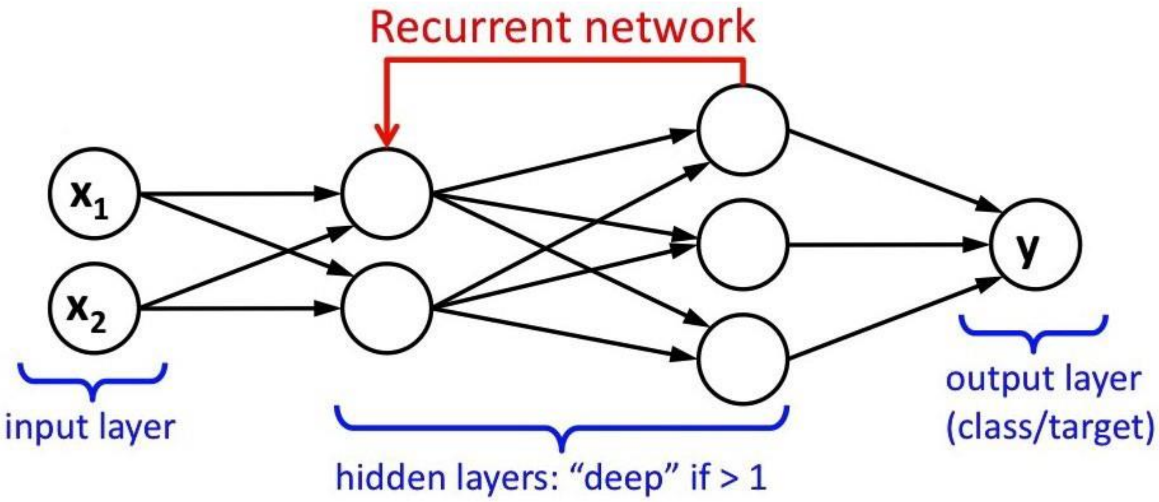


Figure 3.2 Recurrent Neural Network

3.1.3. Convolutional Neural Networks (CNN)

A CNN is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions, convolution and pooling functions are used as activation functions [12]. Figure 3.3 illustrates a Convolutional Neural Networks.

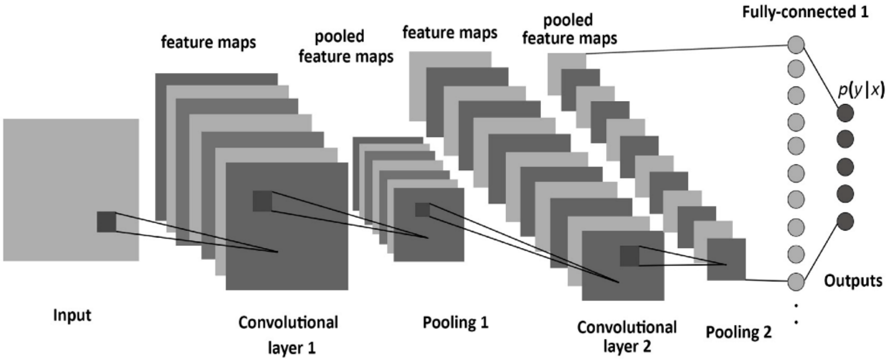


Figure 3.3 Convolutional Neural Networks

3.1.4. Adam Optimizer

Adaptive Moment Estimation (ADAM) is a method that calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients. It can be viewed as a combination of Adagrad, which works well on sparse gradients and RMSprop, which works well in online and nonstationary settings.

Adam implements the exponential moving average of the gradients to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients while staying computationally efficient and having very little memory requirement. Adam optimizer is one of the most popular gradient descent optimization algorithms.

3.2. Background on Feature Extraction

A review has been conducted to figure out best techniques and implementations for extracting features in order to be used by a lip-reading model for translating the hearing-impaired lip signs into actual words either as text or as voice. The following subsections will summarize the findings.

Lip-reading is typically known as visually interpreting the speaker's lip movements during speaking. Experiments over many years have revealed that speech intelligibility increases if visual facial information becomes available. This effect becomes more apparent in noisy environments. Usually, taking steps toward automating this process, some challenges are raised such as co-articulation phenomenon, visual units' type, features diversity and their inter-speaker dependency. Hence it is really important to perfectly decide and engineer the features that will discriminate each word or letter from one another. We elaborate on some of the issues.

3.2.1. Co-articulation Phenomenon

Co-articulation phenomenon is one of the challenges that will appear during taking steps toward automating the lip-reading process, Compared to the conventional audio speech recognition tasks. As stated by Vakhshiteh, Almasganj and Nickabadi (2018) in [13] co-articulation phenomenon can cause the visible speech articulators to be in different positions for the same underlying sound, and thus, the visual features become more dependent to the context (Lan et al., 2012a).

3.2.2. Visual Features

Vakhshiteh, Almasganj and Nickabadi (2018) in [13] stress the importance of choice of features for visual speech recognition purposes, as it is wide. In the literature, several types of visual features are evaluated, which have been commonly categorized as those depending on pixels and those based on the models (Matthews et al., 2002) in [14]. Sunil S. Morade and Suprava Patnaik (2016) in [15] stated that the recognition of lip motion is a difficult task since the region of interest (ROI) is nonlinear and noisy. They also stated that their proposed method in lip reading system used two stage feature extraction model, which is precise, discriminative, and computation efficient. The first stage is 3D Discrete Wavelet Transform (3D-DWT) or 3D Discrete Cosine Transform (3D-DCT) and the second stage is Locality Sensitive Discriminant Analysis (LSDA) to trim down the feature dimensions.

3.3. Background on Image Processing

Image Processing is one important aspect in our project as it is used to detect the lips of the user. Many algorithms are suggested for lips extraction, but in order to dive in the suggested algorithm we will take introduce some important functions used to support our implementation.

3.3.1. Canny Edge Detector

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. John F. Canny developed it in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

The Process of Canny edge detection algorithm can be broken down to five different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

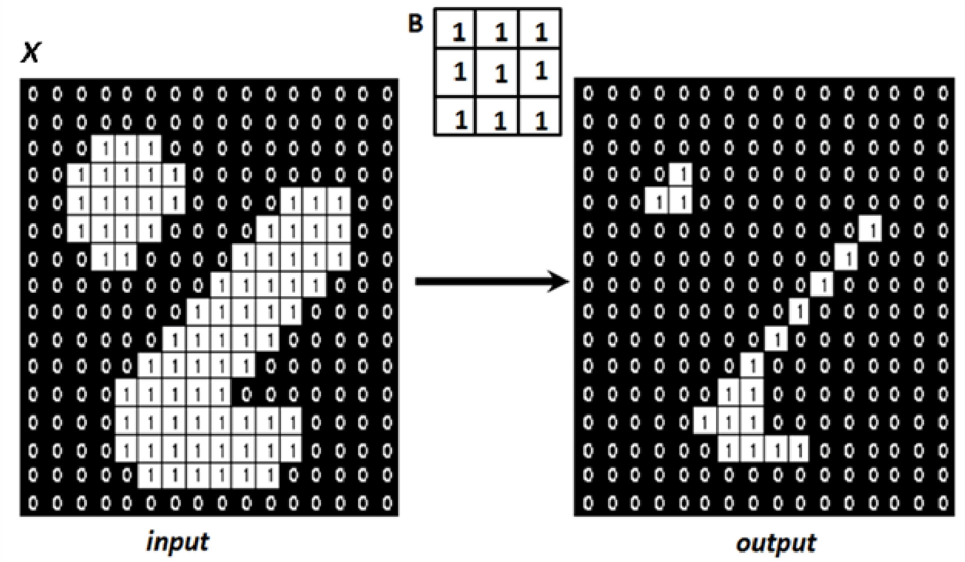
3.3.2. Color Space Functions

Color space functions consist of color space changing functions such as (RGB to Gray – RGB to YCbCr and RGB to HSV) each function was used to support working with different color space in order to obtain more details for the skin mask detected in the algorithm.

3.3.3. Morphological Operations

Morphological operations consist of two main functions, which are erosion and dilation:

- Erosion: The erosion operation usually uses a structuring element for probing and reducing the shapes contained in the input image as shown in Figure 3.4.



X

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

input

B

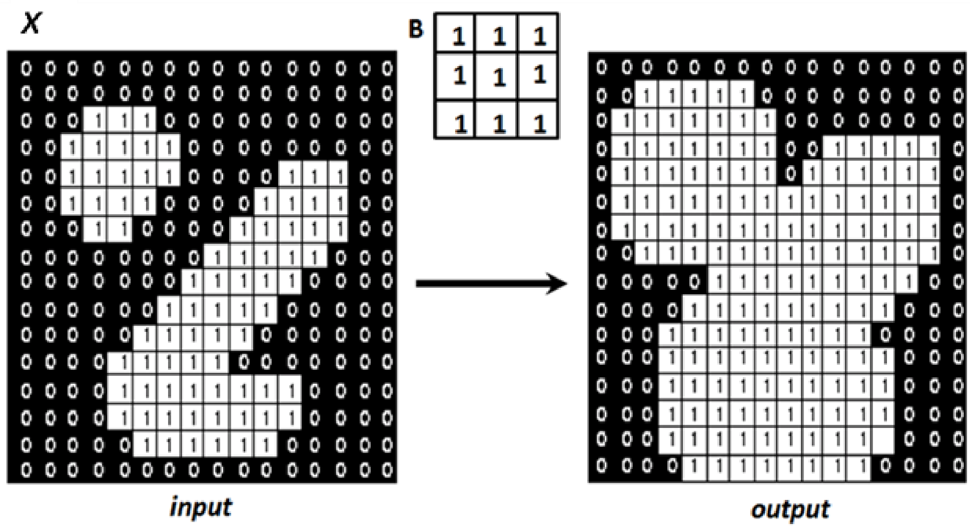
1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

output

Figure 3.4 Erosion operation

- Dilation: The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image as shown in Figure 3.5.



X

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

input

B

1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1		0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

output

Figure 3.5 Dilation operation

Using morphological opening operation (erosion followed by dilation using the same structure) to filling holes in the skin mask that was extracted from the suggested skin detection algorithm and removing unnecessary details such small background objects that are detected by mistake in skin detection algorithm.

3.3.4. Skin Detection Algorithm

Skin color model RGB-HSV-YCbCr for human face detection the first suggested model. This model utilizes the additional hue and chrominance information of the image on top of standard RGB properties to improve the discriminately between skin pixels and non-skin pixels. In this approach, skin regions are classified using the RGB boundary rules introduced by Peer et al. and additional new rules for the HSV and YCbCr subspaces. These rules are constructed based on the skin color distribution obtained from the training images.[16]

When plugging this skin model that was suggested by the authors [16] to our suggested approach using the morphological operations and canny edge detection we reached an accuracy that close to 50% when working with ideal condition images and accuracy of 20% when working with non-ideal images. Unfortunately, this approach is NOT applicable for real life applications due to using static values to address skin tone, which fails to detect some skin tones, it is also highly affected by illumination and dark places with no light.

3.3.5. Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments. Dlib’s open source licensing allows you to use it in any application, free of charge [17].

3.3.6. Haar Cascade Classifier

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001 [17]. It is a machine learning based approach where a cascade function is trained from many positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

1- Haar Feature Selection
2- Creating Integral Images
3- AdaBoost Training
4- Cascading Classifiers

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object. This approach may not be as accurate as Dlib but it is relatively faster than Dlib specially when working in the data preparation, thus we applied Haar cascade in preparation because we can afford to drop few frames since we are working with a large data set.

3.4. Background on Mobile Application Development

Mobile app development is the act or process by which a mobile app is developed for mobile phones. When building a mobile application both a front-end and back-end are needed. The nature of the app and its requirements also make the developer choose between native mobile development and hybrid mobile development as shortly elaborated.

3.4.1. Hybrid Mobile Development

Hybrid mobile development is the process of creating a mobile application that works on multiple systems using a single code base. Much like a hybrid car that works on both gasoline and electricity using the same motor and main components. Technologies used include React Native, Xamarin, Ionic, and Flutter.

Advantages of hybrid mobile application development are:

a) Single code base
b) Lower cost
c) Simpler to build and test
d) Easier to maintain
e) Faster delivery time

3.4.2. Native Mobile Development

Native mobile development is the process of creating a mobile application for a specific type of system. Development is restricted in using that system's programming language and technologies but this results in better code compatibility and better performance than hybrid approaches. Technologies for iOS include Objective-C and Swift. Technologies for Android include Java and Kotlin.

Advantages of native mobile application development are:

a) Better performance
b) Data protection
c) Overall functionality
d) Customer experience
e) Comprehension

3.4.3. Our Mobile Requirements and Choices

Our requirements for Lipify are a fast, stable, and easy to use technology that enables us to easily maintain its code base, work on many platforms as possible, and enable us to access device features. So, our obvious choice was to use a hybrid mobile development technology. As mentioned before, the contenders are React Native, Xamarin, Ionic, and Flutter.

From our research online we concluded the following from these references [19] and [20]:

a) Flutter is 15% faster than Swift for iOS development

b) Flutter can also make web apps

c) Flutter uses the Dart programming language which is based on C/C++.

d) Flutter beats all others in offering the best User Interfaces.

e) Flutter is the newest of them all but is heavily supported by Google.

f) Flutter needs dedicated files for both Android and iOS.

g) Flutter shows the aftereffect of any code changes instantly using hot reload making development faster and easier.

Due to all these advantages and the fact that the disadvantages of Flutter won't greatly affect the final product, we chose Flutter.

3.5. Comparative Study of Image Processing

In this section we give a comparative, classified short literature review of the publications which we chose from them two main publications that showed promising results in detecting face and lip of human face. These papers used pure image processing with color space manipulation.

3.5.1. Face Detection Using Skin Tone Segmentation

Ajith Abraham[16] implemented an algorithm on detection multiple human faces using human skin tone only. In this paper the author used RGB-HSV-YCbCr to detect human face. The Skin region segmentation was performed using combination of RGB, HSV and YCbCr subspaces, which demonstrated evident discrimination between skin and non-skin regions.

The experimental results showed that this approach in modeling skin color was able to achieve a good detection success rate. Ajith Abraham proposed using RGB, HSV, YCbCr Color space with direct threshold on each space to extract skin mask which introduced common issues:

1- Regions are fragmented and often contain holes and gaps.

2- Occluded faces or multiple faces of close proximity may result in erroneous labeling

3- Extracted skin color regions may not necessarily be face regions. There are possibilities that certain skin regions may belong to exposed limbs (arms and legs) and also foreground and background objects that have a high degree of similarity to skin color.

To solve these issues, the author proposed using Morphological operations to close gaps and holes that was introduced by threshold and then calculating bounding box of each connected component and by setting a condition on ratio between length and breadth to eliminate the detected limbs and keep face only.

On the other hand, the author restricted the algorithm to work only with people that have a certain skin tone as well as the assumption that the person in the image mainly show the less skin area than face or else the algorithm will fail and results in erroneous labeling, which made the accuracy of the algorithm to decrease when working with a single face detection and with similar colored background to the human skin.

3.5.2. Human Face Detection Using Skin Color Segmentation and Watershed Algorithm

Abdulganiyu Abdu Yusuf and Fatma Susilawati Mohamad and Zahraddeen Sufyanu[27] implemented an algorithm on detecting human faces with lips localization using human skin tone.

In this paper the authors used RGB-YUV-YIQ-HIS-YCbCr to detect human face and lip localization. The suggested framework demonstrated an efficient face detection using skin color segmentation to get face candidates. The authors proposed using RGB, YUV, YIQ, HIS, YCbCr Color spaces with direct threshold on each space to extract skin mask that will be considered a face candidate.

After getting face candidates the authors proposed using watershed algorithm which consist of two stages:

- The first stage is to isolate the biggest connected white areas and merges small isolated white areas to the background.

- The second stage is to separate the facial features from the background. At this stage, the facial features such as a pair of eyes and nose can be seen clearly.

After the watershed algorithm stage which causes loss of lip region the authors proposed using lip map equation that was introduced by Ali, A. & Sedigheh in their paper "Robust Component-based Face Detection Using Color Feature (2011)".

This algorithm had detection rate of up to 97.22% that was obtained from their own data base which is more accurate and more efficient than the proposed algorithm in 3.5.1

The authors restricted the algorithm to work with images from standard databases in order to enable comparison with benchmark results, adding to that it faces a problem for determining the range and proper color model.

3.6. Comparative study on Feature Extraction

In the **feature extraction** part Vakhshiteh, Almasganj and Nickabadi (2018) in [13] have proposed hybrid features in a proposed well-designed arrangement of certain function blocks. As the final classifier, a single DBN with a well-designed topology, inside a DBN-HMM structure, was established and employed for all speakers.

Sunil S. Morade & Suprava Patnaik (2014) in [15] have proposed 3D-DWT, Discrete Wavelet transformation, method to extract features out of the lip region. 3D-DCT, Discrete Cosine Transform, was also proposed. Sunil and Suprava stated that 3D-DCT is a simpler technique than 3D-DWT, yet it is as efficient as 3D-DWT.

The proposed **visual feature** set by Vakhshiteh, Almasganj and Nickabadi (2018) in [13], so-called hybrid features is as follows:

each video file is converted to a sequence of frames, in which the speaker's body is covered from the shoulders to the head. Each frame is pre-processed, so as to extract only the ROI encompassing the mouth. For this purpose, Active Shape Modeling algorithm (Cootes *et al.*, 1995) is used, via which face landmarks are obtained. Face detection is made by tracking the landmarks placed on the border of face. Eighteen landmarks are positioned on the inner and outer lip contours. Similarly, by tracking these points, an exact bounding box (B.B) of the lips is obtained. Each determined

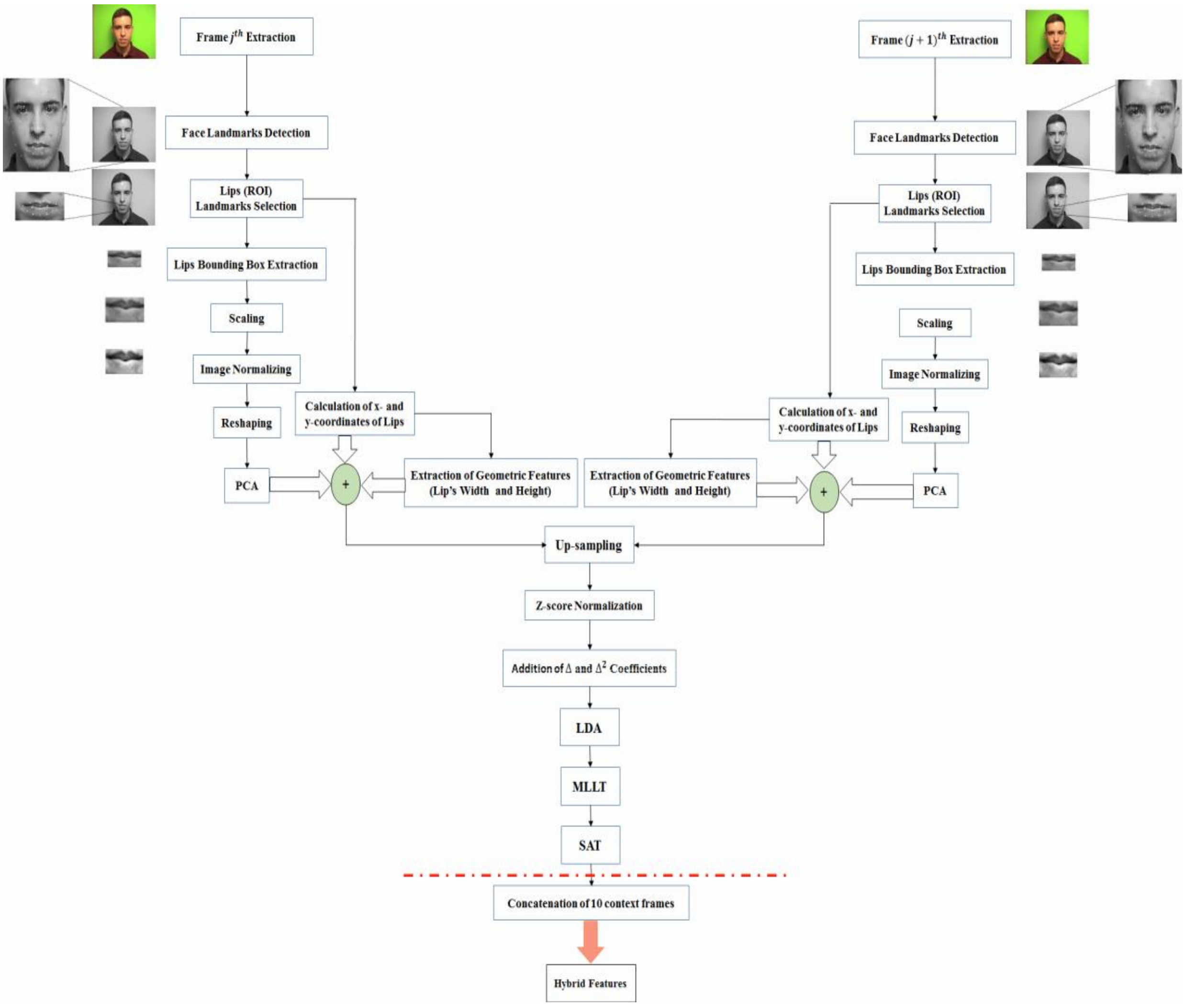

Frame j^{th} Extraction
Face Landmarks Detection
Lips (ROI) Landmarks Selection
Lips Bounding Box Extraction
Scaling
Image Normalizing
Reshaping
PCA
Calculation of x- and y-coordinates of Lips
+
Extraction of Geometric Features (Lip's Width and Height)
Extraction of Geometric Features (Lip's Width and Height)
+
Calculation of x- and y-coordinates of Lips
Frame $(j+1)^{th}$ Extraction
Face Landmarks Detection
Lips (ROI) Landmarks Selection
Lips Bounding Box Extraction
Scaling
Image Normalizing
Reshaping
PCA
Up-sampling
Z-score Normalization
Addition of Δ and Δ^2 Coefficients
LDA
MLLT
SAT
Concatenation of 10 context frames
Hybrid Features

Figure 3.6 The schematic of the hybrid feature extraction process.

mouth B.B is rescaled to 25x40 pixel size, and finally is converted to a 1000-dimensional feature vector. This vector is further transformed to a 32-dimensional vector, using the PCA algorithm. In this way, the so-called appearance-based features are created. From the (x, y) lip coordinates, the width and height of the mouth are calculated and used as the geometric features. Concatenating the x- and y-coordinates with the geometric features, a 38-dimen-sional vector is created, which we call it the shape-based feature set. These features are combined with the appearance-based features, making a 70-dimen-sional feature vector. This is illustrated in Figure 3.6 The schematic of the hybrid feature extraction process.. As the final classifier, a single DBN with a well-designed topology, inside a DBN-HMM structure, was established and employed for all speakers. Experiments were applied on the CUAVE database which yields accuracy of 61.5% [13].

The proposed visual features set by Sunil S. Morade & Suprava Patnaik (2014) in [15], was tested against CUAVE database for recognition rate of lip reading of individual candidate using 2D-DWT as a feature extraction method and BPNN classifier with M parameter (Momentum term) equals to 0.001, it scored an average accuracy of 73.43% [21].

3.7. Implemented Approaches for Image Processing

Our suggested implementations is made of two different approaches, one uses three different color spaces with more complex functions 3.7.1 and the other uses only one color space with mathematical operation 3.7.2.

3.7.1. YCbCr/HSV/RGB skin Model Approach:

Our requirements for image processing phase were easy, fast and accurate lip detection algorithm that could be to be compared with the two libraries that we mentioned in 3.3.5 Dlib and 3.3.6 Haar cascade classifier, so our choice was to use the algorithm by Ajith Abraham [16], but with some modifications to the algorithm to solve the problem faced in the algorithm by observing the output if the proposed algorithm and record the errors occurred due to the single face detection that we want to obtain from this algorithm.

Our proposed fixes that we used in the implementation:

1- For the erroneous labeling, we proposed using contours and selecting the largest contour area.

2- For the missing lip detection method, we proposed using the human face aspect ratio, which state that the human face is divided into three parts, this solution provided the detection of the lip's region but in a large region that might affect the feature extraction. To solve this problem we applied trial and error to get a general aspect ratio for the mouth region that concentrate entirely on the mouth region only with minimum error as shown in section 5.4.2.

Due to these solutions, the proposed method in section 5.4.2 using algorithm proposed by Ajith Abraham[16] was able to meet the requirement of our product and could substitute to the proposed method in section 5.4.5 but at lower accuracy.

3.7.2. RGB/Normalized RGB skin Model Approach:

Unlike our first approach 3.7.1, that dependent on three color spaces for the skin model. In this approach we used the single-color space that was suggest by Abdellatif Hajraoui and Mohamed Sabri[22].

Using only the suggested RGB skin model we managed to extract the same skin region as our first approach but with less details that did not affect the output of the algorithm.

After extracting the skin region we took a different approach from the one that was suggest by Abdellatif Hajraoui and Mohamed Sabri[22] and Ajith Abraham[16] by observing the output of each step from the first approach we came with an easier and more accurate implementation that majorly count on mathematical calculations.

Chapter 4: Dataset Preparation and Preprocessing

Before getting into our system architecture as detailed in the next chapter, this chapter gives a general block diagram and then explains how we dealt with the data obtained from the dataset to work with our system. It includes subsections data visualization, data preparation, feature extraction, word processing, and preprocessing.

4.1. Data Visualization

The first step in the project cycle, is to visualize the data to know what we have and what to do. We have some data that is composed of several videos of volunteers that were asked to pronounce certain sentences, as shown in Figure 4.1.

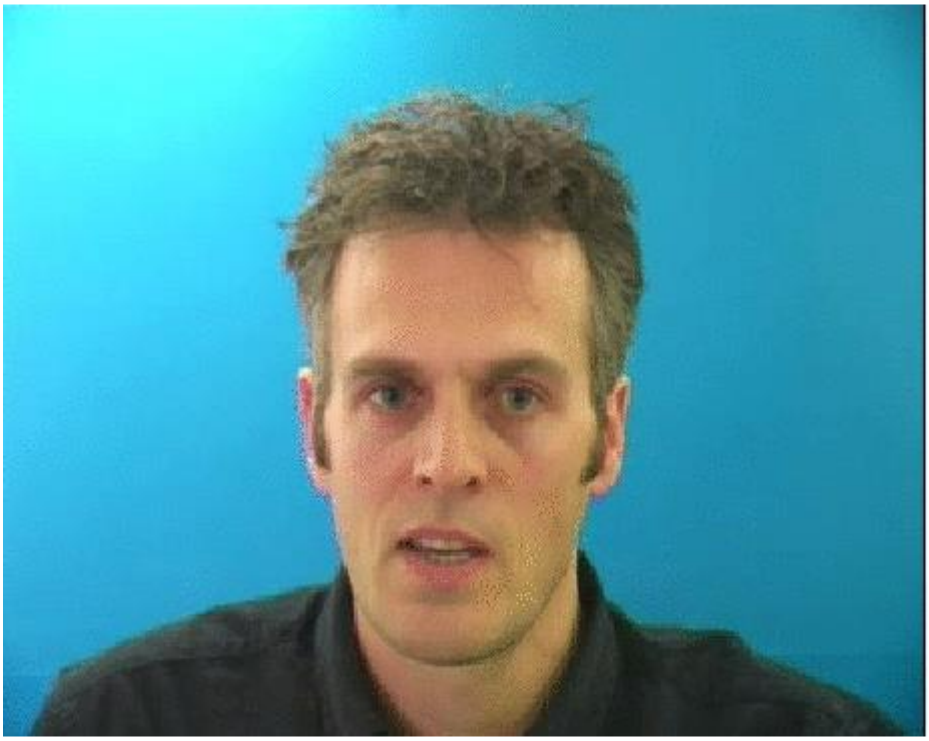


Figure 4.1 Sample of Data Set

The videos that are used as our data set are clear enough to have a clear view of the lips, which is a necessary condition in all our data. The videos do not suffer from blurring or any type of noise that might require image processing in order to become a valid data.

4.2. General Block Diagram of the System

Input
An audio less video of a person talking
Output
Text prediction of what the person said

Figure 4.2 General System Block Diagram

4.3. Data Preparation

Data was not ready to be used right away in the analysis process. We have to make some transformations and approximations in order to make it usable.

4.3.1. Capturing Lips

For lip capturing, we reviewed several approaches to use in order to capture the lips from our data set that will be used to train and test the network model:

1. Skin Segmentation Algorithms to detect skin region of speaker then applying contours on segmented parts to detect face followed by using human face aspect ratio to detect lips region.

2. Applying Haar cascade on the face to extract face interest points which includes the mouth interest points that is used in feature extraction phase.

Figure 4.3 shows one of the dataset samples where the lips were detected. While Figure 4.4 shows a block diagram showing the data preparation steps.

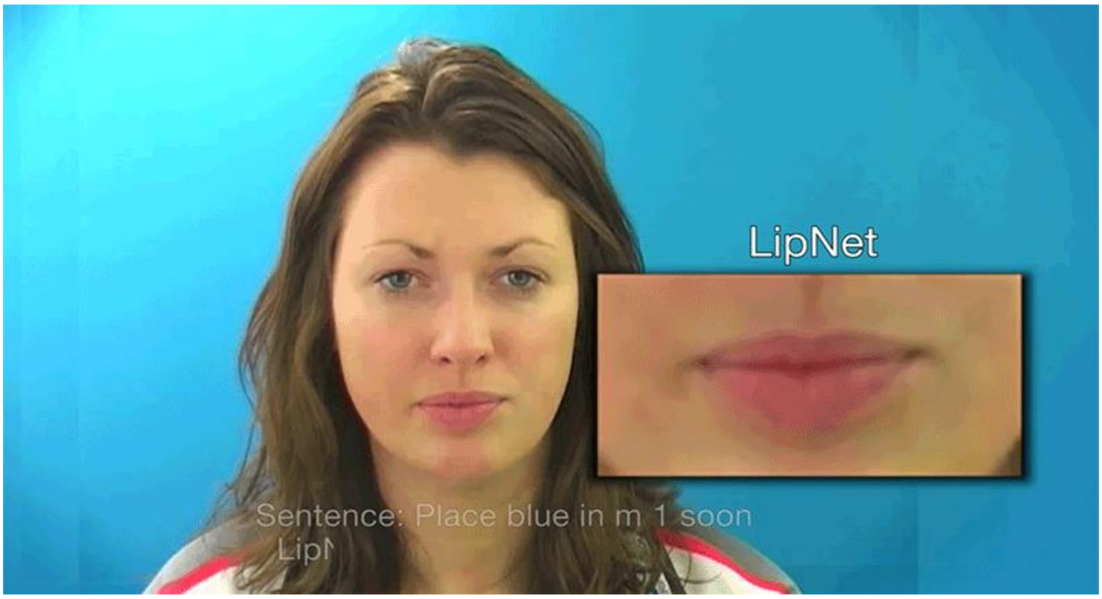


Figure 4.3 Detected Lips

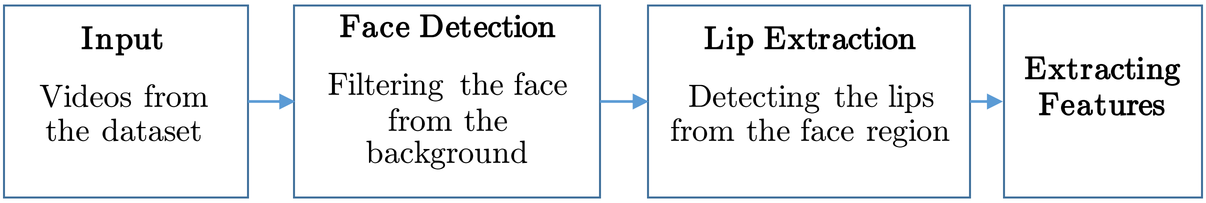

Input
Videos from the dataset
Face Detection
Filtering the face from the background
Lip Extraction
Detecting the lips from the face region
Extracting Features

Figure 4.4 Data Preparation

4.4. Features Extraction

In feature extraction there have been typically variant approaches including Active Shape Model (ASM) which is a top-down approach introduced in 1995 [23], Active Appearance Model (AAM) which combines the same (top-down) shape analysis used for ASM tracking with a bottom-up statistical model of gray level appearance [14], and 3D-DCT & 3D-DWT techniques.

A major challenge faced by the lip-reading model is that a human produces less visual variation as compared to acoustic phonetics. E. Petajan [24] experimented on lip-reading to enhance speech recognition. Matthews et al. [25] worked on DCT, DWT and PCA image transform methods. The image transform methods have been compared with Active Appearance Model (AAM) and found that Image transform methods have highest recognition accuracy as compared to AAM. Following the lead of [15], features will be extracted from the lips region using a 3D-DWT or a 2D-DWT for simplicity.

Discrete Wavelet transformation (DWT) is originally used to compress images. It is simply removing high frequencies out of the image. A linear high and low pass filters are convoluted with the image in both vertical and horizontal directions. After convolution, the image is down sampled by two then convoluted with a high and low filter in parallel.

Linear Discriminant Analysis (LDA) is further used to reduce the dimensionality of the features. LDA tries to figure out a new space with the order of the groups to be categorized in which it maximizes the distance between means of the classes and, minimize the variation within each class.

4.5. Video Capturing

For the data capturing part, the plan is to use this project on mobile devices thus we are using the mobile camera through a custom made up by us to capture a short video of the speaker and send the frames of the video which are 25 frames per second to the backend to enter the data preparation phase. Figure 4.5 shows this process.

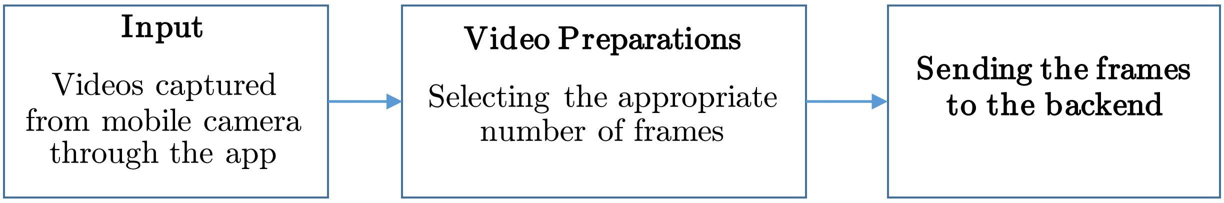

Input
Videos captured from mobile camera through the app
Video Preparations
Selecting the appropriate number of frames
Sending the frames to the backend

Figure 4.5 Video Capturing Steps

4.6. Word Processing

We use the GRID corpus to train and evaluate our model because it is sentence-level and has the most data. The sentences are drawn from the following simple grammar:

Command (4) + color (4) + preposition (4) + letter (26) + digit (10) + adverb (4)

Where the number denotes how many word choices there are for each of the 6-word categories.

The categories – respectively – consist of:

a) {bin, lay, place, set}
b) {blue, green, red, white}
c) {at, by, in, with}
d) {A → Z}
e) {0 → 9}
f) {again, now, please, soon}

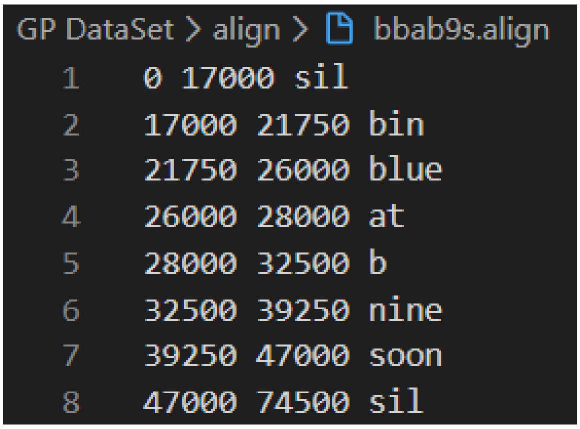
This yields 64000 possible sentences. For example, two sentences in the data are "set blue by A four please" and "place red at C zero again". We then apply a **word to index transformation** on each word category in our grammar structure to be able to match the network's output with the actual labels.

For Example:

Command Category: {"bin" → 0, "lay" → 1, "place" → 2, "set" → 3}

4.7. Dataset Preprocessing

As stated in the last section that our dataset consists of videos of each speaker uttering ***1000 different statements*** following our dataset's grammar. The scope of our project is to work on word level instead of sentence level to provide more flexibility to our users, so the first challenge was to use the provided align files, shown in Figure 4.6, to segment. Figure 4.7 shows the dataset structure after segmenting.



GP DataSet > align > bbab9s.align

```
0 17000 sil
17000 21750 bin
21750 26000 blue
26000 28000 at
28000 32500 b
32500 39250 nine
39250 47000 soon
47000 74500 sil
```

Figure 4.6 Align file example

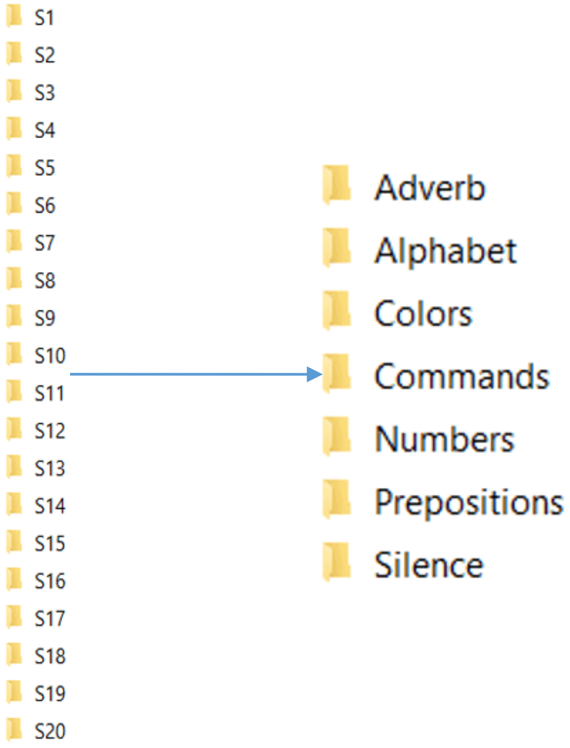


Figure 4.7 Dataset structure after segmentation

After segmenting the dataset, a problem arose, the videos did not contain a constant number of frames and each video was different. We searched for various solutions to this problem including:

a) Replicating frames of the video to reach the desired number (30 frames)
b) Adding dark frames to reach desired number
c) Using Silence videos for the same speakers to fill the missing frames

Our chosen approach is to use the third option, in each video there was a **small moment of silence** before and after each sentence. Therefore, after extracting them we used each speaker's video with their respective category to add the silence frames in 3 positions, illustrated in Figure 4.8:

a) Case 1: Before uttering the word.
b) Case 2: After uttering the word.
c) Case 3: Both before and after uttering the word.

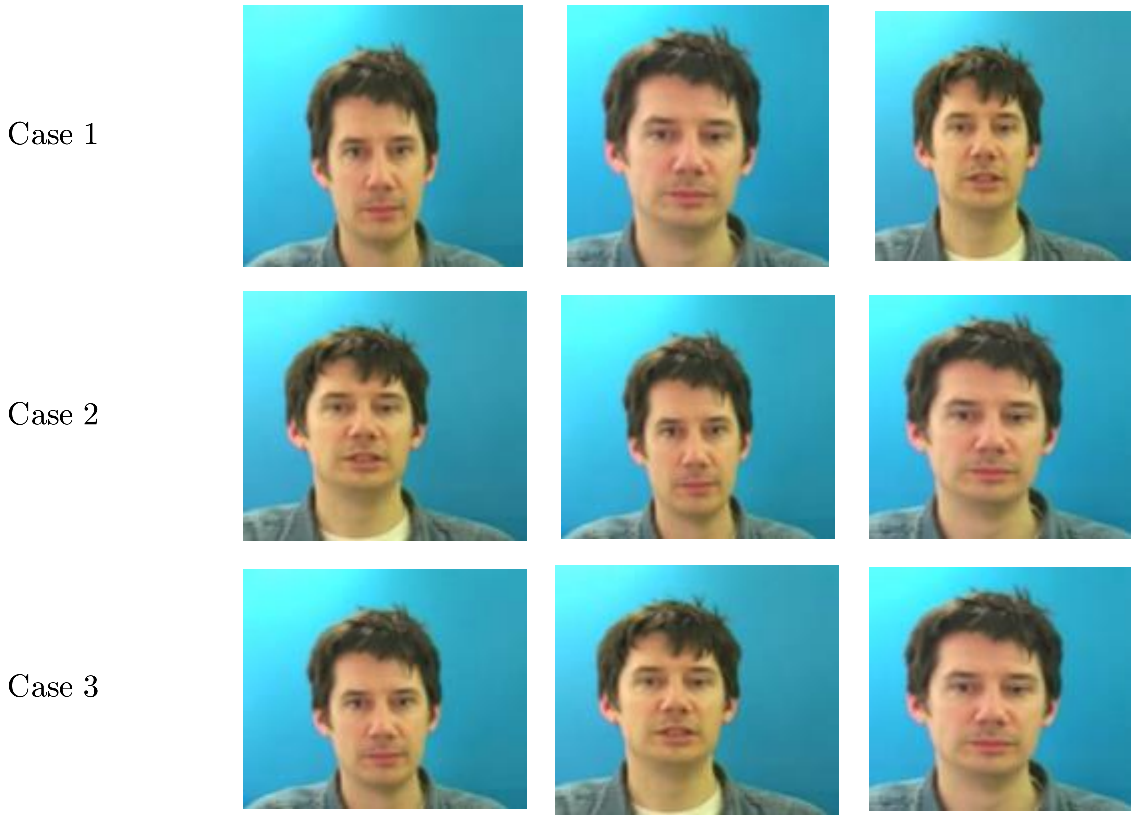


Figure 4.8 Sentence uttering cases

4.8. Dataset Concatenation

After pre-processing our dataset and normalizing the videos to 30 frames each, we started extracting the mouth regions from the videos to prepare our data for CNN training. We started by extracting video frames and then extracting the mouth region, in this part we compared the performance of various extraction techniques:

Table 4.1 Comparison between facial extraction techniques

	DLIB	YCbCr + HSV	Haar Cascade
Time to extract per video	72 Seconds	140.8 Seconds	2.5 Seconds
Accuracy in extraction	~100%	~55%	~82%

We found that using DLIB produced an accuracy of almost 100% but made the segmentation process extremely slow and not viable for a large dataset. That is why we decided to go with using Haar Cascades in the segmentation process as it produces acceptable results in the shortest time. As shown in Figure 4.9 and Figure 4.10:

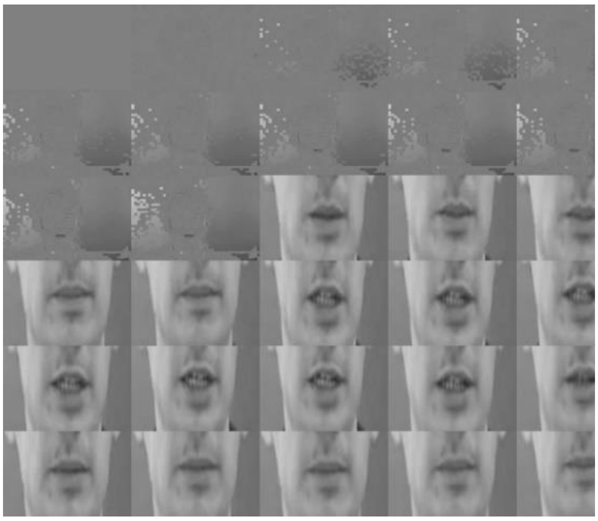


Figure 4.9 Inaccurate segmentation using Haar Cascade

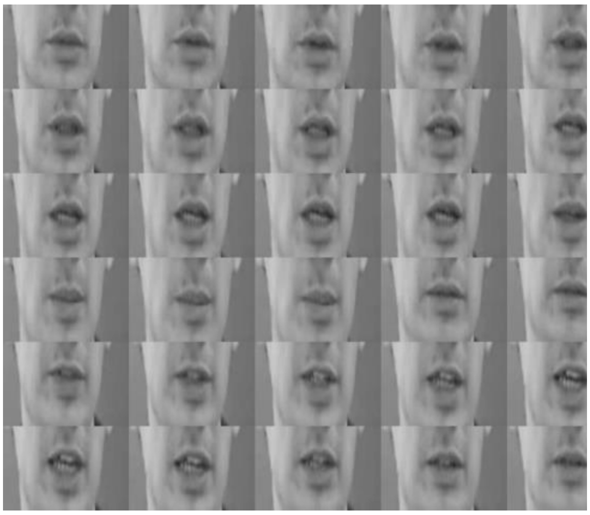


Figure 4.10 Accurate segmentation using Haar Cascade

Chapter 5: System Design and Architecture

This chapter describes the project in full details. We try to answer the questions: "What has been done?" and "How it has been done?" So, in this chapter, every module in our project is discussed in details starting with our system architecture block diagram.

5.1. Overviews and Assumptions

Lipify is built to help all hearing-impaired around the world and since English is the most used language in the world, Lipify assumes that the user knows the English language. We also assume that the user is familiar with using a mobile phone's camera and can easily capture videos.

5.2. System Architecture Block Diagram

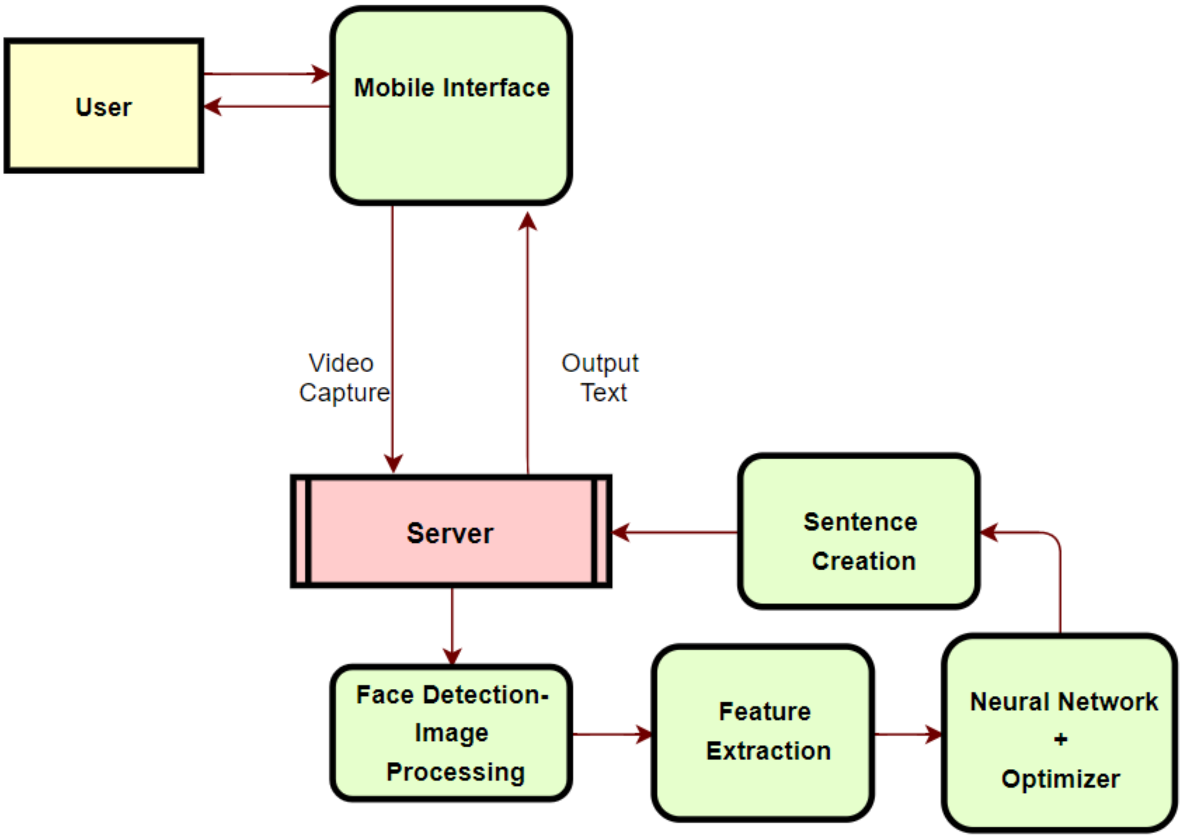


Figure 5.1 System Architecture Block Diagram

5.3. Feature Extraction Module

The proposed implementation for this module will be built from scratch, not following the deep learning approach. It is mainly based on generating statistics out of mouth coordinates to specify what the letter most likely is.

Deep learning approach will make use if the CNN to extract the most effective features needed later to train the learning module on the given grammar. On the other hand, it is expected to build from scratch a module using some points on lips, around eight points, and calculate the vertical distance between each other, comparing the distance between the given points in this frame and previous frame to extract well specifying features.

5.4. Image Processing Module

Our proposed Implementation for the lips detection module contains four main approaches, which are:

5.4.1. First Implementation: Dlib

Dlib contains many algorithms. In our project we are only using one of them which is a part of the images processing, high quality face recognition algorithm. The reason for choosing this approach is the high quality and accuracy that Dlib face recognition provides, we reached an accuracy of 98% when detection user mouth. Dlib also provides the interest points which consists of 68 points, which we use only 20 points in our mouth detections.

Dlib face detections that uses 68 points to define face landmarks by using a machine learning algorithm to detect face and extract face landmark and apply these points to it. Figure 5.2

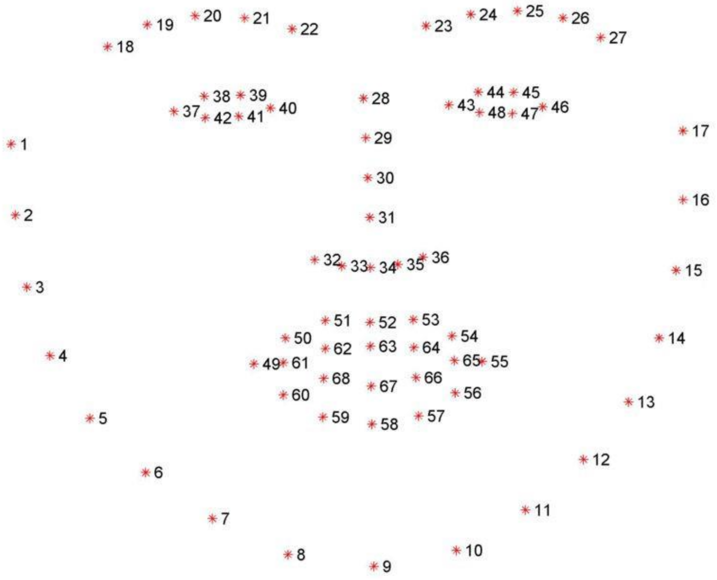


Figure 5.2 Dlib 68 face landmark points

This method is highly accurate and fast since it uses a machine learning algorithm instead of pure image processing algorithms and it is not affected by external factors such as light intensity like the second method that we will discuss later on.

On the other hand, this approach faced a problem when working with data preparation because it took considerable time just to finish detecting mouth of a single speaker file which lead us to use a different approach when working with data preparation and detection.

5.4.2. Second Implementation: Skin Segmentation for Lips Detection

Unlike the Dlib implementation this method uses purely image processing techniques by manipulating the color space channels and using edge detection algorithms such as:

a) RGB, HSV, YCbCr color space
b) Canny Edges Detector
c) Contours
d) Connected components
e) Gaussian filter
f) Morphological operations (erosion, dilation)
g) Thresholding

This method directly manipulates the image and does not depends on any machine learning algorithm. Adding to that, it shows in depth the lips extraction method using pure image processing. Its result is shown in Figure 5.3.

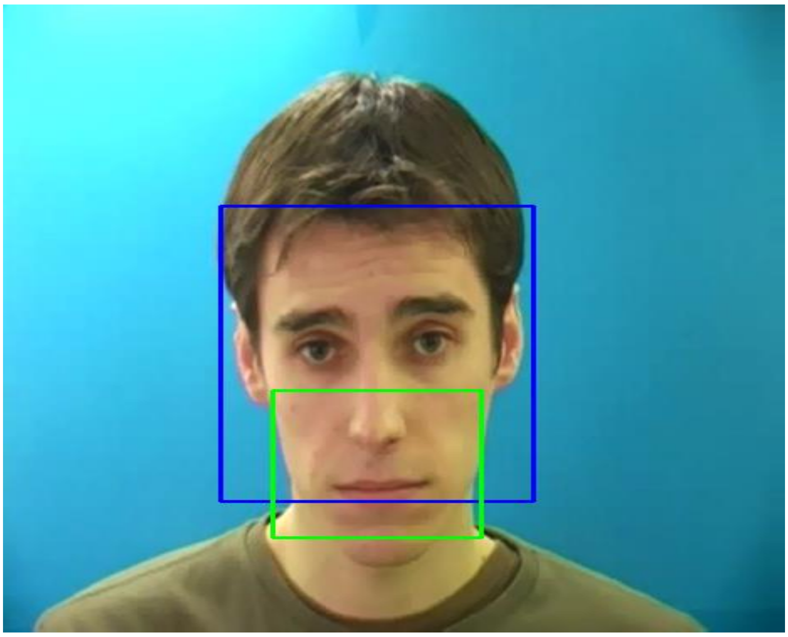


Figure 5.3 Detecting Lips

Even though this method is simpler than using the first implementation that we proposed, it had many problems:

- RGB color space is affected by light intensity which causes major errors in the skin segmentation phase
- Any human/background that have a color degree similar to the human skin color might be wrongly segmented
- Contours depends on area to discriminate between different segmented skin regions which cause an error when selecting the correct segments.
- Mouth area contains error margins

A major problem that affect the detection greatly is the mistakenly detected area in the skin segmentation phase which we managed to solve this problem using contours, this solution solved the problem of the wrong detected areas and increased the accuracy of the face detection to 50% in ideal conditions. Lastly, in order to detected mouth, we used face aspect ratio from the detected face and cut face according to the ratio using trial and error approach.

We feel that this method will impose a major problem when using it on noisy environment (non-ideal), so after researching we found that Haar cascade was more suitable and almost error free also faster than this method, adding to that Haar cascade is the most used face detection which an accuracy up to 98%.

5.4.3. Third Implementation: RGB Skin Segmentation for Lips Detection

Unlike the YCbCr/HSV skin segmentation implementation this method uses less color spaces and less complex functions by manipulating only the one color space and using edge detection algorithms and mathematical calculations such as:

- RGB/Normalized RGB Skin Model [22]
- Filling Holes in the skin mask using morphological operation (Opening)
- Canny Edges Detector
- Using contours to segment the binary mask
- Repeat until no more segments are available or face segment detected:
    a. Counting number of white pixels in each segment
    b. Selecting segment with the highest white pixel count
    c. Counting number of holes in that segment
    d. If number of holes is 3 or more then the segment contain face, if not remove this segment from the segment list and repeat from (a)
- From the detected segment apply face to mouth aspect ratio to extract mouth ROI

This method directly manipulates the image and does not depends on any machine learning algorithm. Adding to that, it provided an accuracy close to 85% which is much higher than the implementation used in 5.4.2. Its result is shown in Figure 5.4.

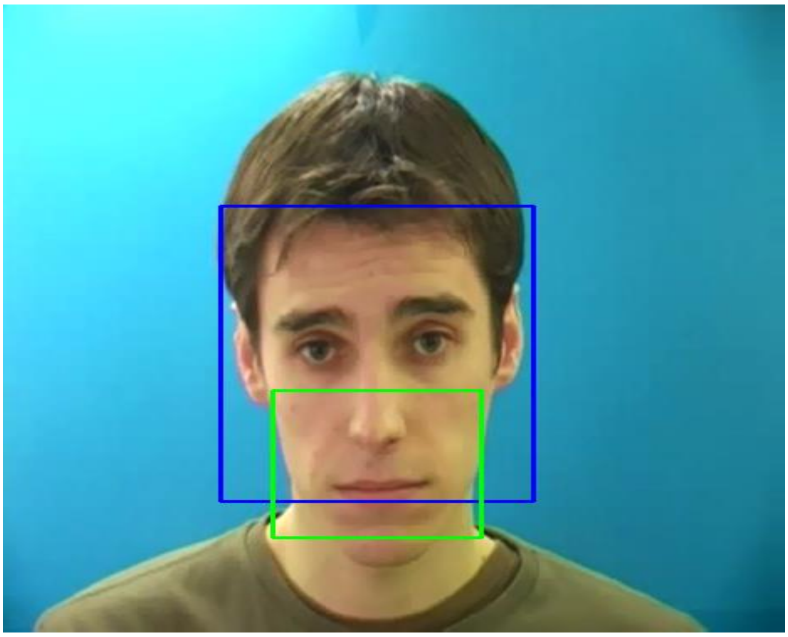


Figure 5.4 RGB model output

5.4.4. Fourth Implementation: Skin Segmentation for Lips Detection Implemented Functions

The previous implementation contains already made functions that enhanced the performance of the skin segmentation algorithm that was implemented in order to extract the mouth ROI. On the other hand, in this implementation we will work with pixel level by implementing and comparing the output of already implemented function with our written function and assess why would we have preferred working with the already implemented functions.

The following functions were used in the project and were implemented by hand to show the difference between the two methods, for simplicity we worked with simple yet detailed images to show that difference in depth.

1) Canny Edge Detector

While OpenCV supported the use of canny edge detector using simple function call with only deciding on upper and lower boundaries of the threshold, our function dive in depth by implementing the parts of canny which are (RGB to Gray Color transformation – Gaussian Blur – Sobel edge Detection - non maxima Suppression and hysteresis threshold).

After implementing canny we found the already implemented function runs faster and have higher accuracy for detecting the edges but contain more noise as shown in the images below using the same parameters for each method. A comparison is shown in Figure 5.5.

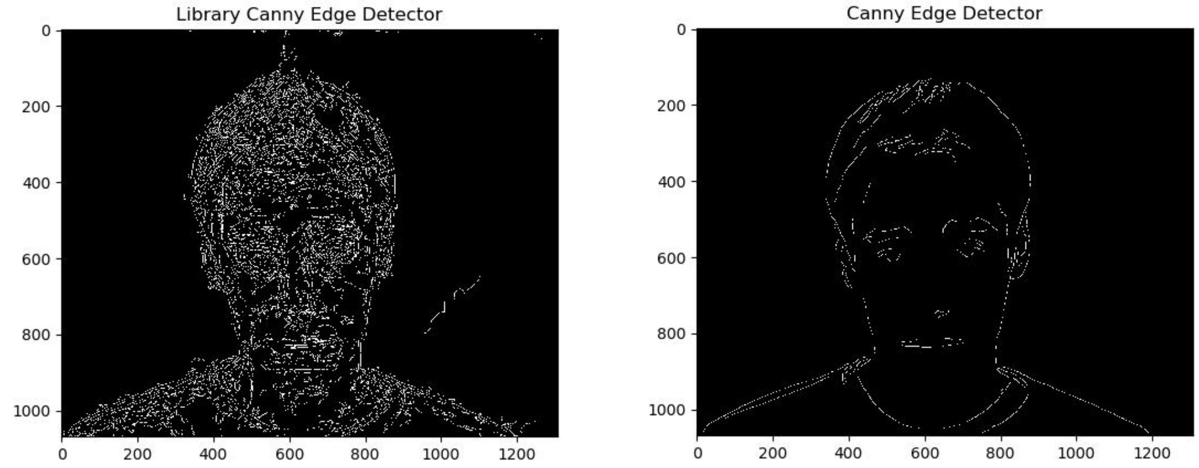


Figure 5.5 Canny Library vs Canny Coded

2) Color Space Functions

Color space functions consist of color space changing functions such as (RGB to Gray – RGB to YCbCr and RGB to HSV) each function was used to support working with different color space in order to obtain more details for the skin mask detected in the algorithm.

Using the already implemented functions changing color space was only faster than the coded functions and the coded function depended on static functions which was used to transform from RGB to HSV which was the main reason for choosing the already implemented function, we will show the comparison between the RGB to HSV as demonstration of the outputs of both functions.

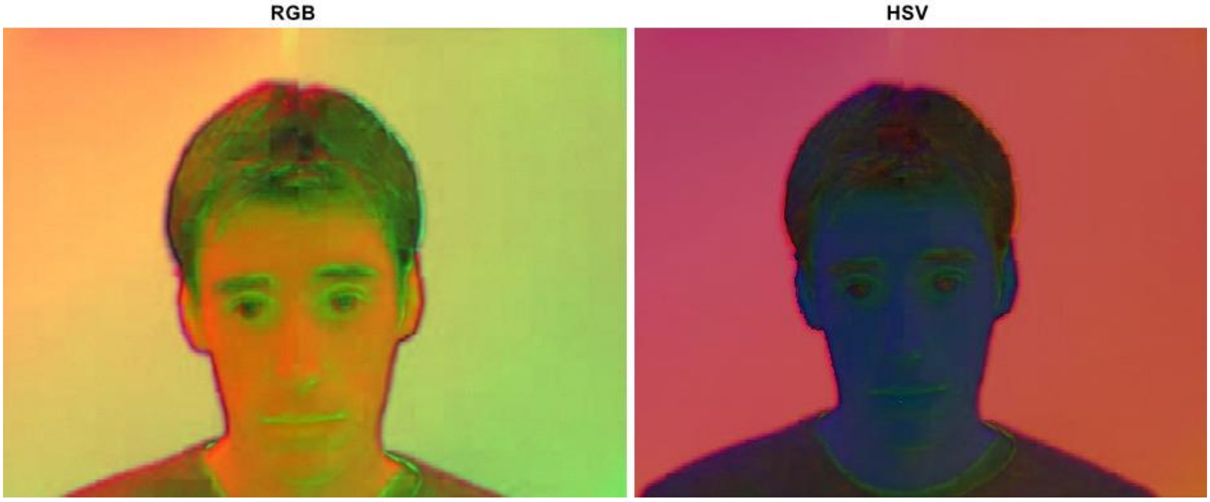


Figure 5.6 RGB to HSV

As observed from Figure 5.6, changing the equation values result in different outputs for the hand coded function which results in negative effect when working with our data set due to having different skin tone speakers so working with static function will results in massive failure in detection mouth ROI using coded function thus it was better to work with the already implemented function.

3) Morphological operations:

Morphological operations consist of Erosion and dilation of image using a specific structure element, this method allow us to emphasize on important parts of our image using the dilation operation and discarding irrelevant parts using erosion operation, combining both operations in different orders results in two important operations which are opening (dilation then erosion) and closing (erosion then dilation).

Using the already implemented functions (opening – erosion – dilation) resulted in faster processing of the image and using less computational power, also the output of image is smoother than the handmade function because it detected automatically the appropriate structure element for each frame.

On the other hand, both approaches (handmade – already implemented) functions provided results with the same accuracy, and adding to the accuracy the handmade functions allowed us to added padding to our image, as demonstration of the difference between both approaches. Using opening operation which will demonstrate the significant differences as shown in Figure 5.7.

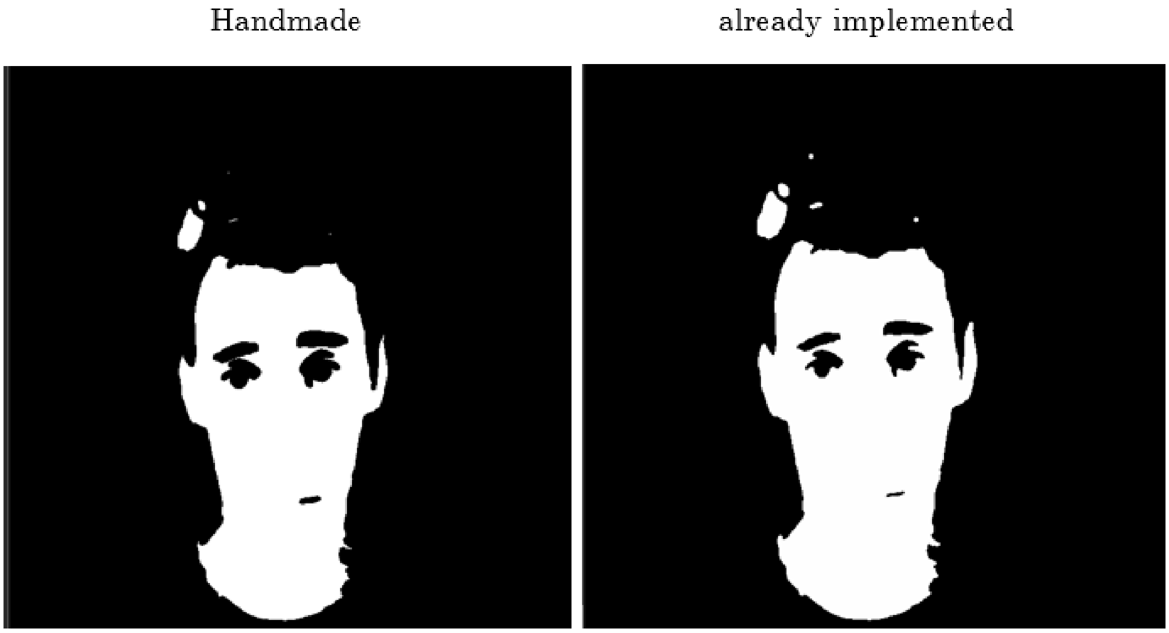


Figure 5.7 Opening operation comparison

We notice that the already implemented function as mentioned before is smoother than the handmade but the handmade function using padding of size 1 on the image we managed to remove some irrelevant parts in the image. Even though the handmade function is better since it removes the smaller parts we are sacrificing the speed of our implementation for the sake of removing parts that will be handled in the canny edge detection and area calculations, thus we decide to use the faster implementation which is the already made function.

4) Sobel Filter:

Used in image processing, particularly within edge detection algorithms where it creates an image emphasizing edges, computing an approximation of the gradient of the image intensity function. At each point in the image, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical as shown in Figure 5.8.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 5.8 Sobel filter operators

The output gradient of the Sobel filter was exactly the same as the already implemented function and the time taken by each approach was slightly different where we could consider it neglect the difference. Adding to speed the output gradient was clear showing all edges perfectly as shown in Figure 5.9.

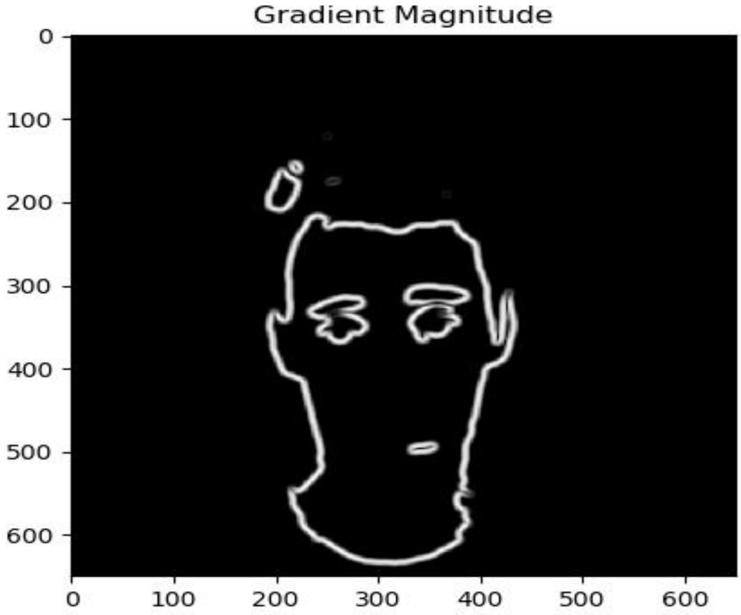


Figure 5.9 Sobel filter gradient output

Applying Sobel filter on the image is one of the steps of canny edge detector, which is the main reason for implementing it in this module, but when applying our handmade canny edge detector that was mentioned before, we found by observing the output that we could the only difference between the canny and Sobel was the non-maxima suppression and the hysteresis threshold, thus for making our pixel level implementation to work faster and avoid further delay we decided to drop the non-maxima suppression and hysteresis threshold and stop the canny at Sobel filter output which resulted in the following output Figure 5.10.

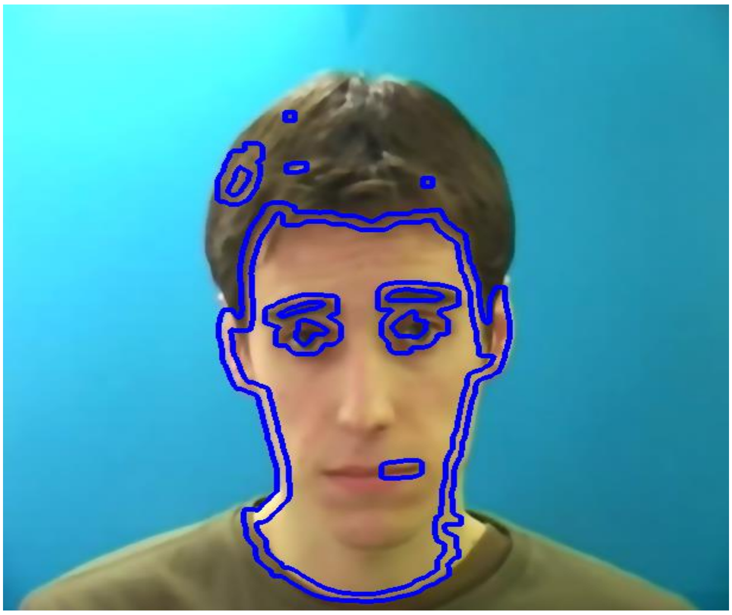


Figure 5.10 Canny modified output

Due to using steps from canny which is a second order filter instead of using Sobel filter directly, we ended up with an inner edge that should be suppressed by the hysteresis threshold and non-maxima suppression that we decide to drop from our steps and only stop at Sobel output, we solved this inner edge by applying area calculation on each closed region and only kept the large closed area, this way we managed to enhance the speed of the pixel level implementation while keeping the accuracy of our output the same.

5) Gaussian Filter:

A Gaussian filter is a linear filter. It's usually used to blur the image or to reduce noise. If you use two of them and subtract, you can use them for edge detection. The Gaussian filter alone will blur edges and reduce contrast.

The main reason for using the Gaussian filter is to smooth the edges in order to be able to detect only strong edges in the canny edge detection. That is mainly for the purpose of detecting the skin mask boundary. For the Gaussian filter phase, we decide to go with the already implemented function due to the performance of the already implemented function was better than the handmade function as shown in Figure 5.11.

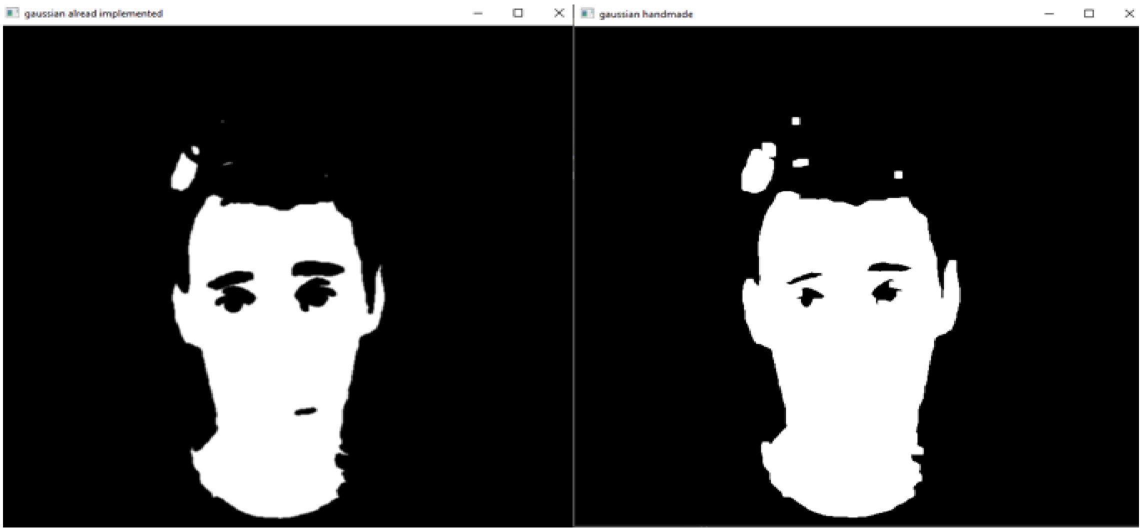


Figure 5.11 Comparison of Gaussian filter

6) Module Output:

After replacing the previous function with our custom handmade functions, the output of the module as shown in (Figure 5.12) was exactly as the already implemented functions which proved that by cutting the hysteresis threshold and non-maxima suppression and applying the area calculation was a successful method to decrease computation. However, unfortunately, due to working with matrices (images) ***the delay of the handmade module was considerably larger than already implemented function module which is the reason for us to drop the handmade model.***

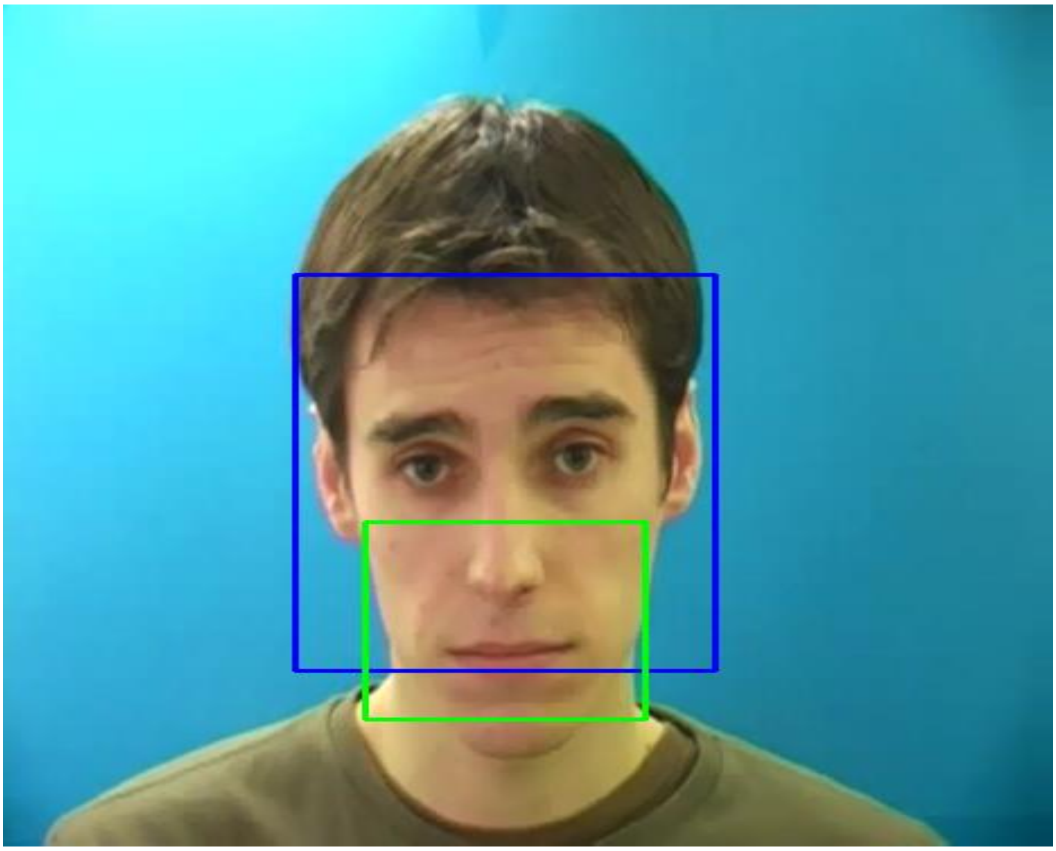


Figure 5.12 Handmade Module Output

5.4.5. Fifth Implementation: Haar Cascade Classifier

The main proposed implementation for the mouth detection module for data preparation and user mouth extraction. Using Haar features as shown in Figure 5.13, integral images and AdaBoost classifier to detect face and mouth as mention in.

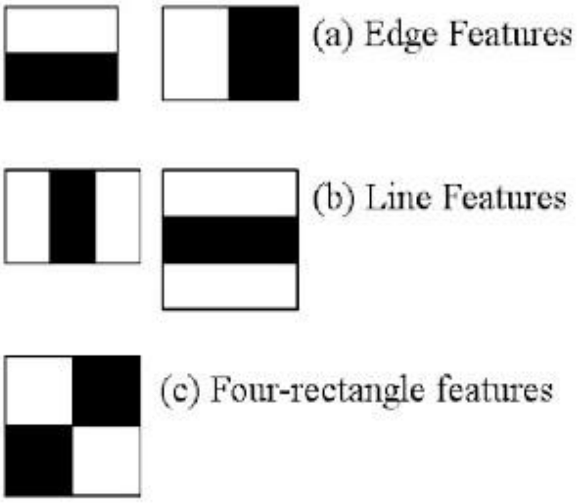


Figure 5.13 Haar Features Used for Face Detection

Unlike Dlib approach, this method is faster in extracting face and mouth but less accurate than Dlib as showing in Figure 5.14. Which is why this method was chosen in data preparation phase because we can afford to drop few frames in order to obtain faster data preparation for our learning model.

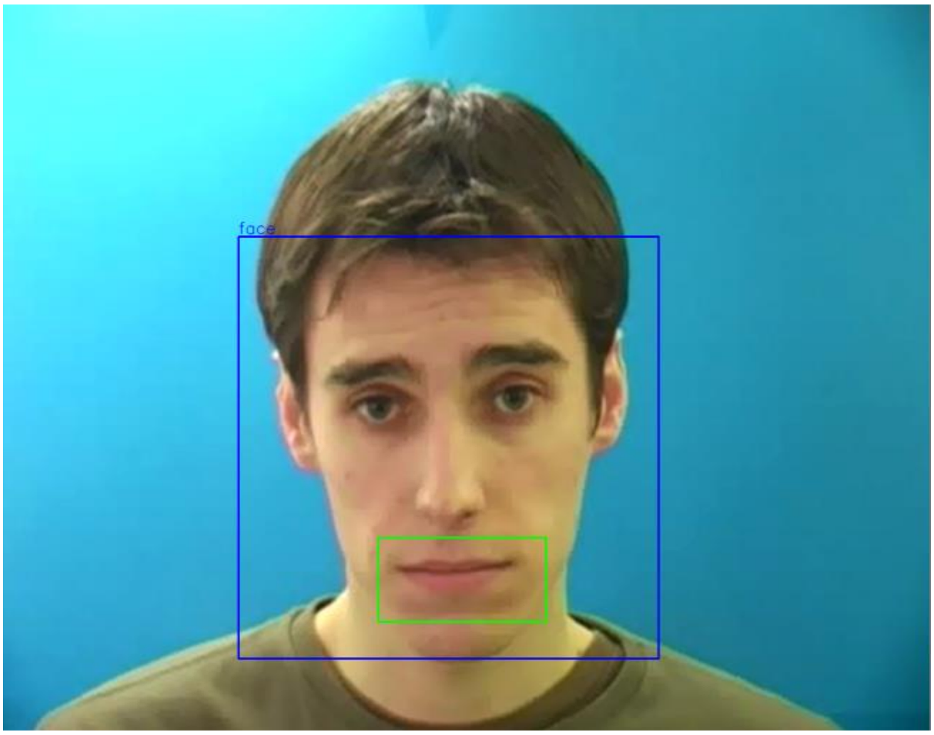


Figure 5.14 Mouth and face detected

5.4.6. Sixth Implementation: Hybrid Module

After discussing and comparing the four previous implementations, we proposed a sixth implementation that combine between the speed of the Haar cascade classifier and the Dlib accurate prediction. The Haar cascade classifier is the proposed method for both of data preparation and user mouth ROI detection, also the Dlib accurate mouth detection that was mainly proposed at first for extracting user mouth ROI but had to be replaced with Haar cascade because it caused a significant delay to the overall system performance.

As a result, this hybrid module is mainly used as ideal module for the mouth ROI extraction, purely for showing the ideal mouth detection method that suffers from failure rate close to zero.

The proposed working method for the module is simple, first we run the Dlib module that was explained in 5.4.1. If the Dlib fails then we simply run the Haar cascade module that was also explained in 5.4.5. If both methods fail, then we resize the whole frame to the mouth ROI dimensions and then send it to the feature extraction module.

This method is robust to failures because if all methods then resized sent frame might give us at least a one feature that could be used from it, thus nothing is lost in this method.

The output of the hybrid module will be the same in case of Haar cascade and Dlib as shown in Figure 5.14, and in case of failure it will be the frame itself as shown in Figure 5.15.

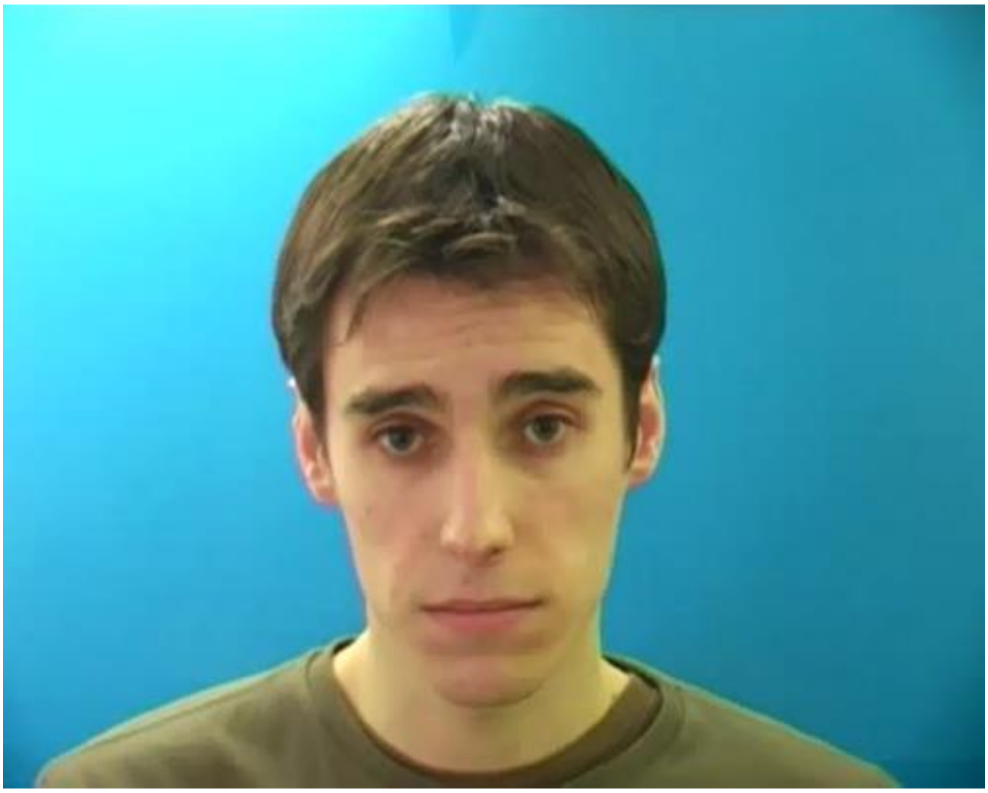


Figure 5.15 Hybrid module failure output

5.5. Learning Module

Our proposed Implementation for the learning module contains two main approaches, which are:

5.5.1. Deep Learning using CNNs and RNNs

The proposed method relies heavily on using Gated Recurrent Networks (GRUs) to make our network memorize the sentence structure so far, but it has a limitation that it will make our solution only capable of recognizing whole sentences while making saying only words and numbers impossible.

We feel that this will impose a great limitation on our target audience, but again our project is highly dependent on the Dataset and based on our research. We found that LipNet [26] Paper has a suitable implementation that was tested against our same dataset and showed an amazing accuracy rate of 90% as opposed to professional lip readers who achieved 55%, the paper's proposed implementation is shown in Figure 5.16.

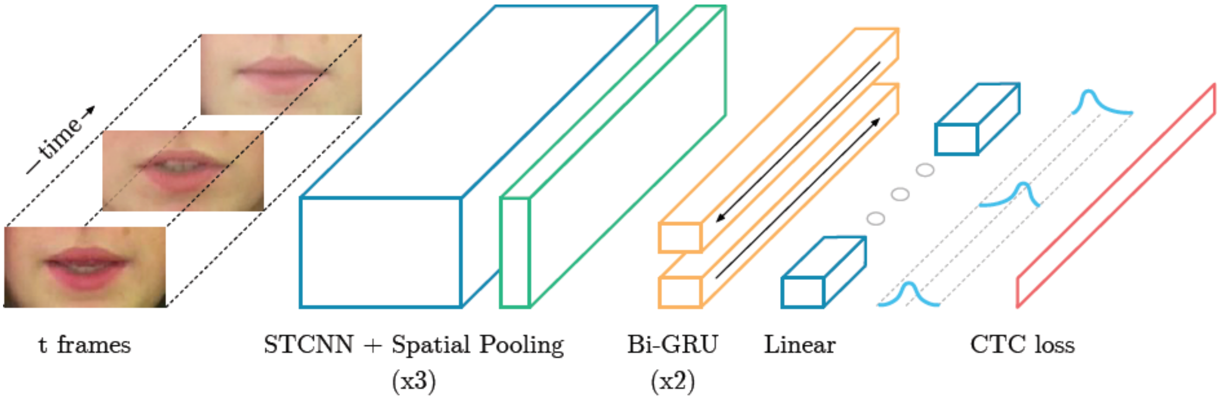


Figure 5.16 LipNet's Proposed Architecture

5.5.2. Artificial Neural Networks

The second proposed solution is based on our particular dataset (GRID corpus), since we know that our grammar consists of 6 main categories which are:

Command (4) + color (4) + preposition (4) + letter (25) + digit (10) + adverb (4)

Then we thought of creating 6 independent networks where each network handles only one category of our grammar structure. The idea is to give our users the flexibility they need to communicate with other people freely and not just restrict them to one grammar structure.

For example, a user can choose to utter just a color, a command, or any combination of the categories, thus providing them with more degrees of freedom to shape the sentence as they please based on their preferences. The proposed implementation of a single neural network will look similar to what is illustrated in Figure 5.17.

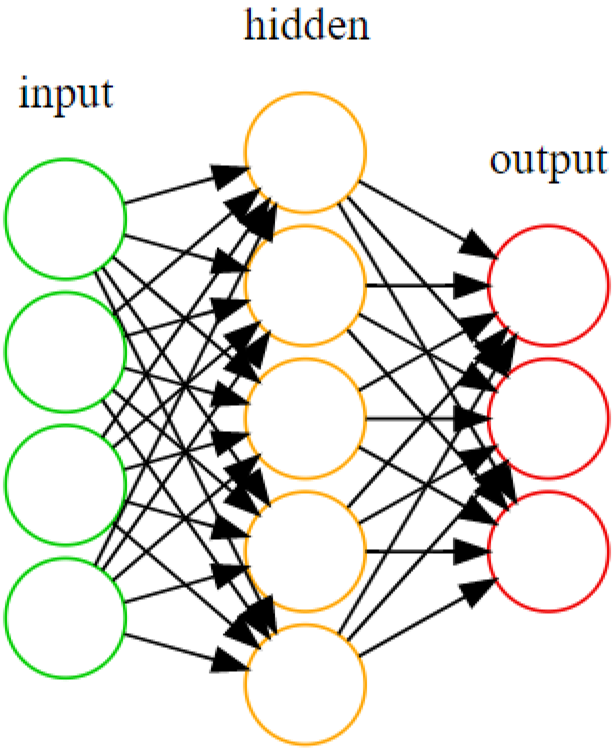


Figure 5.17 Multi-Class Classification Neural Network

5.6. Mobile Interface and Server Module

This module explains what we use as Lipify's user interface and server and why we chose each.

5.6.1. Mobile Interface

Our implementation for Lipify's interface with the user will be a hybrid mobile application made with Flutter, which is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. This will allow us to build both an Android and iOS application while writing the code once for both platforms. Flutter uses the Dart programming language, which is very similar to C++ in syntax.

Our choice of Flutter came after searching and comparing with many other hybrid mobile application development environments. Lipify's requirements are so simple to run a user's device so we did not need a native mobile application approach. After some more digging, Flutter scored better than most other solutions it even beat Swift, which is the native approach for building iOS apps.

The mobile application will be responsible for capturing videos of the user talking, sending the video to the server for analysis, and then showing the predicted text back again to the user. The application will be built from scratch.

5.6.2. Server

As for the server, we are currently sending the video using HTTP multipart requests to the server to be fast in sending and receiving. Currently our server will be one of our powerful laptops and both the mobile device and the laptop will be connected on the same network. The server receives the video and sends back its prediction. The server is written in Node.js® which is a JavaScript runtime built on Chrome's V8 JavaScript engine and its express.js web application framework that provides a robust set of features for web and mobile applications.

Chapter 6: Implemented Architecture

For our project we decided to go with two approaches, firstly, we will use CNNs without feature extraction and the other is to create a NN to predict our labels.

6.1. Adverb CNN Architecture

After reviewing a lot of papers and SOTA architectures, we decided to create our own architecture and to benchmark it against other well-known architectures.

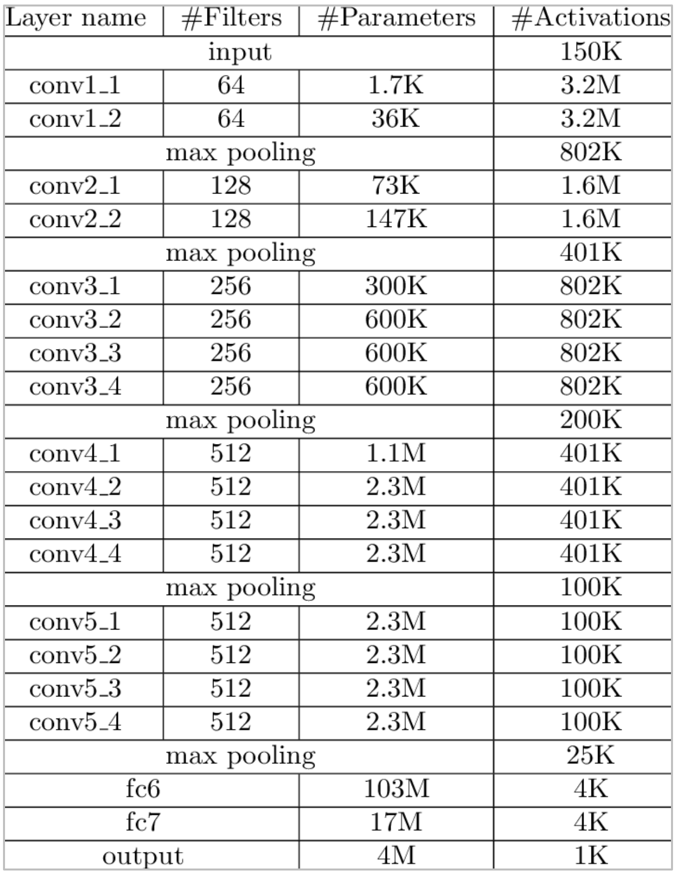
Our project consists of six different CNN architectures, each fine-tuned -Despite their similarities -for one of our speech categories.

We decided to build our Adverbs CNN with the following architectures and compare the results:

6.1.1. Using VGG-19

We utilized an already existing famous architecture like the VGG-19, and trained it on our dataset by ***transfer learning VGG-19 model*** and adding our output layer.

Unfortunately, this did not yield the results that were expected and the model performed poorly on our test set with an accuracy of: ***28.94%***



Layer name	#Filters	#Parameters	#Activations
input			150K
conv1_1	64	1.7K	3.2M
conv1_2	64	36K	3.2M
max pooling			802K
conv2_1	128	73K	1.6M
conv2_2	128	147K	1.6M
max pooling			401K
conv3_1	256	300K	802K
conv3_2	256	600K	802K
conv3_3	256	600K	802K
conv3_4	256	600K	802K
max pooling			200K
conv4_1	512	1.1M	401K
conv4_2	512	2.3M	401K
conv4_3	512	2.3M	401K
conv4_4	512	2.3M	401K
max pooling			100K
conv5_1	512	2.3M	100K
conv5_2	512	2.3M	100K
conv5_3	512	2.3M	100K
conv5_4	512	2.3M	100K
max pooling			25K
fc6		103M	4K
fc7		17M	4K
output		4M	1K

Figure 6.1 VGG-19 Architecture

6.1.2. Building CNN by using Numpy only

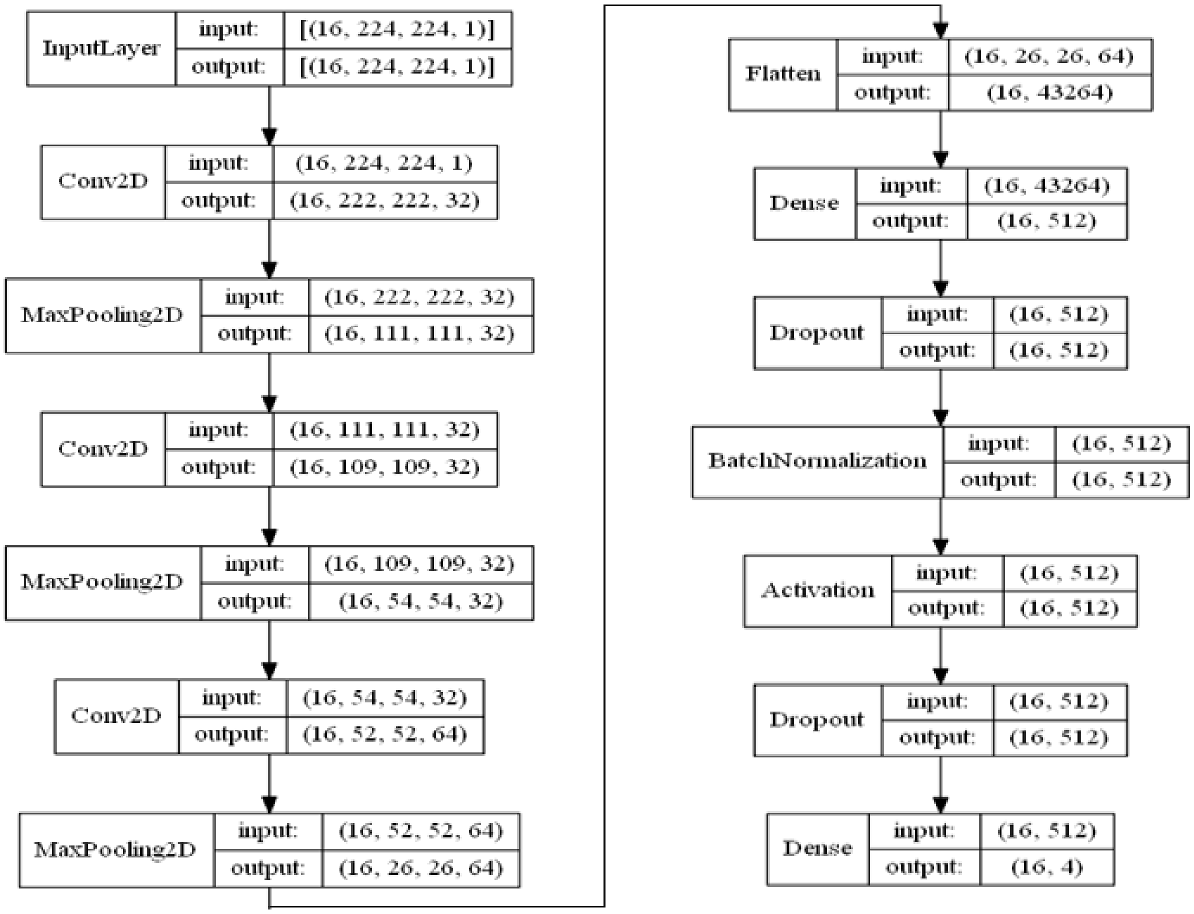
For benchmarking our CNN, we applied the concepts of Convolution, Max pooling and Categorical Cross Entropy using no libraries at all -Except Numpy for Matrices Multiplication.

As expected, the CNN was extremely slow and took ***51 hours*** for ***one full epoch*** on our dataset and after that, the accuracy wasn't actually that good as it was ***26%*** on the test set.

6.1.3. Using our custom architecture

We tried to create our own custom architecture, it took a lot of trials to reach an appropriate architecture capable of differentiating the different words, and we finally reached a suitable architecture as shown in Figure 6.2.

This architecture performed at an outstanding level in comparison to the other two architectures with a test set accuracy of ***92.78%***



InputLayer input: [(16, 224, 224, 1)] output: [(16, 224, 224, 1)]
Conv2D input: (16, 224, 224, 1) output: (16, 222, 222, 32)
MaxPooling2D input: (16, 222, 222, 32) output: (16, 111, 111, 32)
Conv2D input: (16, 111, 111, 32) output: (16, 109, 109, 32)
MaxPooling2D input: (16, 109, 109, 32) output: (16, 54, 54, 32)
Conv2D input: (16, 54, 54, 32) output: (16, 52, 52, 64)
MaxPooling2D input: (16, 52, 52, 64) output: (16, 26, 26, 64)
Flatten input: (16, 26, 26, 64) output: (16, 43264)
Dense input: (16, 43264) output: (16, 512)
Dropout input: (16, 512) output: (16, 512)
BatchNormalization input: (16, 512) output: (16, 512)
Activation input: (16, 512) output: (16, 512)
Dropout input: (16, 512) output: (16, 512)
Dense input: (16, 512) output: (16, 4)

Figure 6.2 Adverb CNN Architecture

6.2. Models' Architecture

After we compared the performance of the adverb model with various implementations in 6.1, we go more in-depth with the performance and architectures of our remaining networks.

6.3. Colors CNN

Our colors dataset consisted of 4 colors: ***blue, green, red, white.*** Our architecture -shown in Figure 6.3– reached an astounding validation accuracy of ***90.68%***

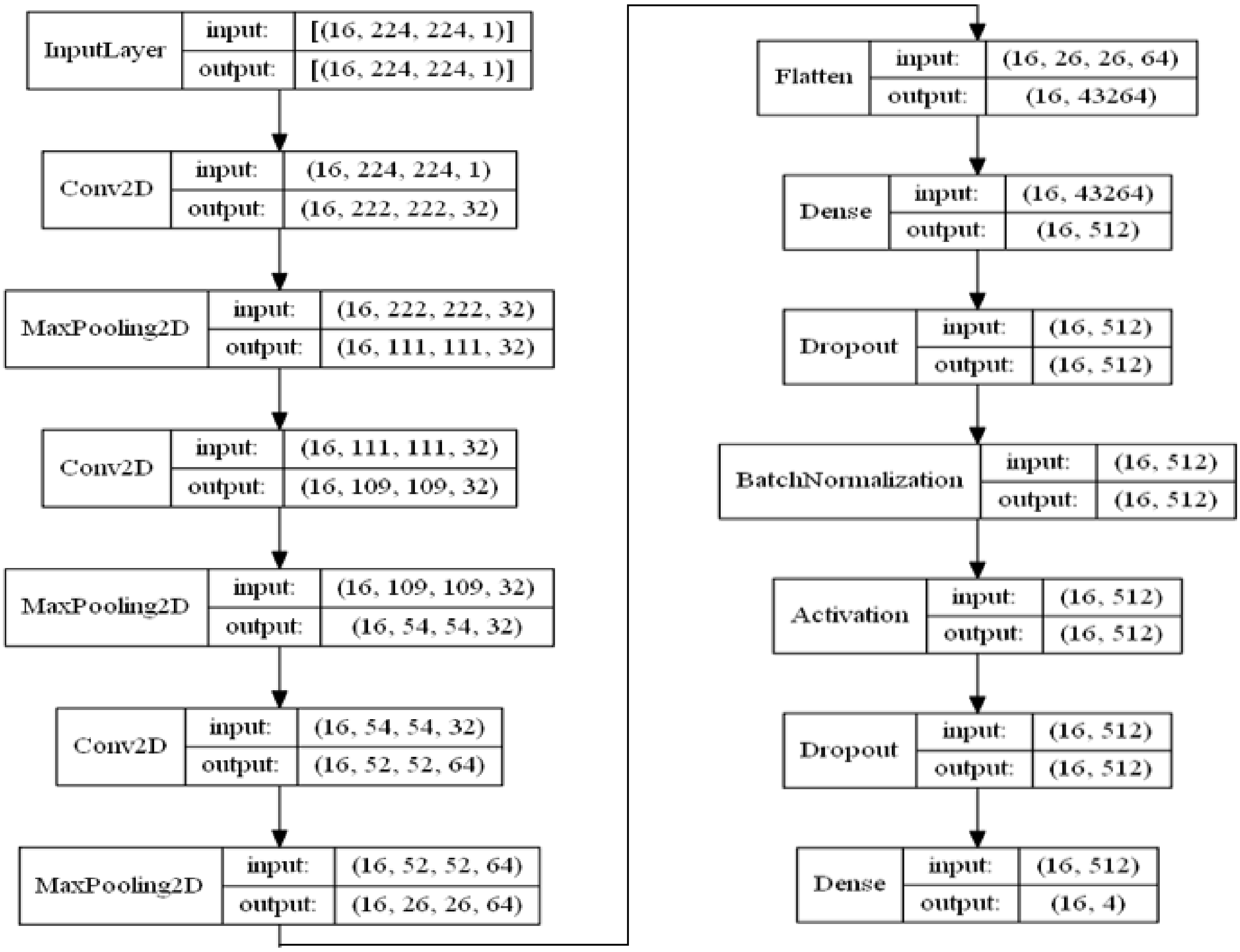
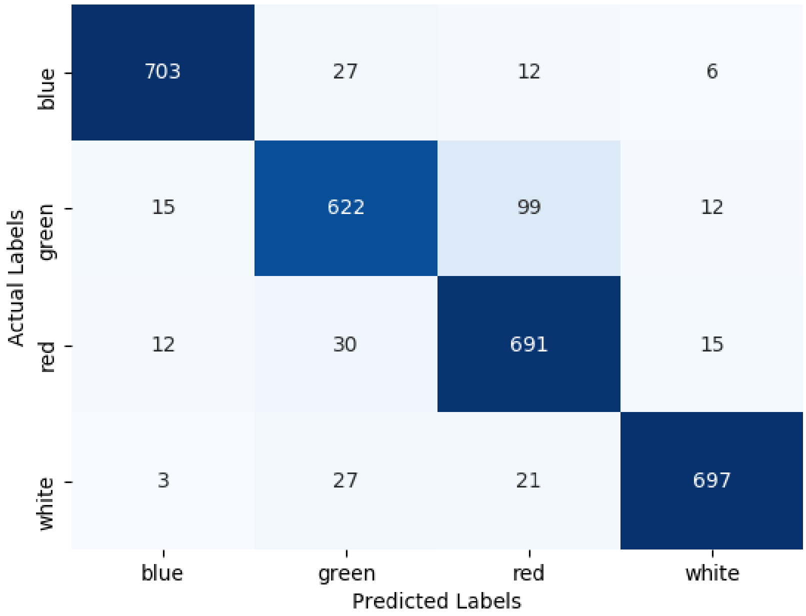


Figure 6.3 Colors CNN Architecture

The CNN had a confusion matrix as shown in Figure 6.4, from it we can observe that the confusion occurs mostly between the colors ***red*** and ***green.***

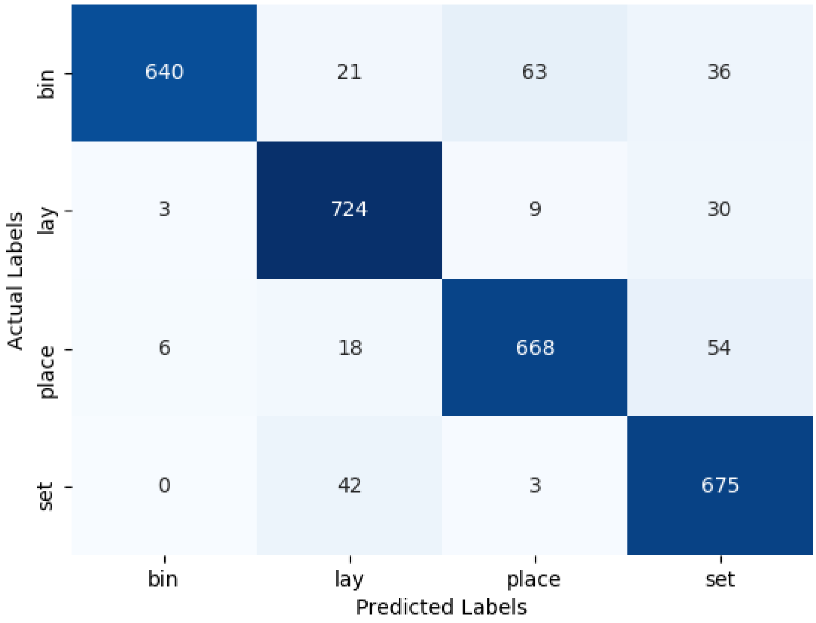


Actual Labels \ Predicted Labels	blue	green	red	white
blue	703	27	12	6
green	15	622	99	12
red	12	30	691	15
white	3	27	21	697

Figure 6.4 Color CNN Confusion Matrix

6.3.1. Commands CNN

Commands CNN's architecture was similar to the architecture presented in Figure 6.3 and reached a validation accuracy of ***90.47%*** with confusion matrix as shown in Figure 6.5:

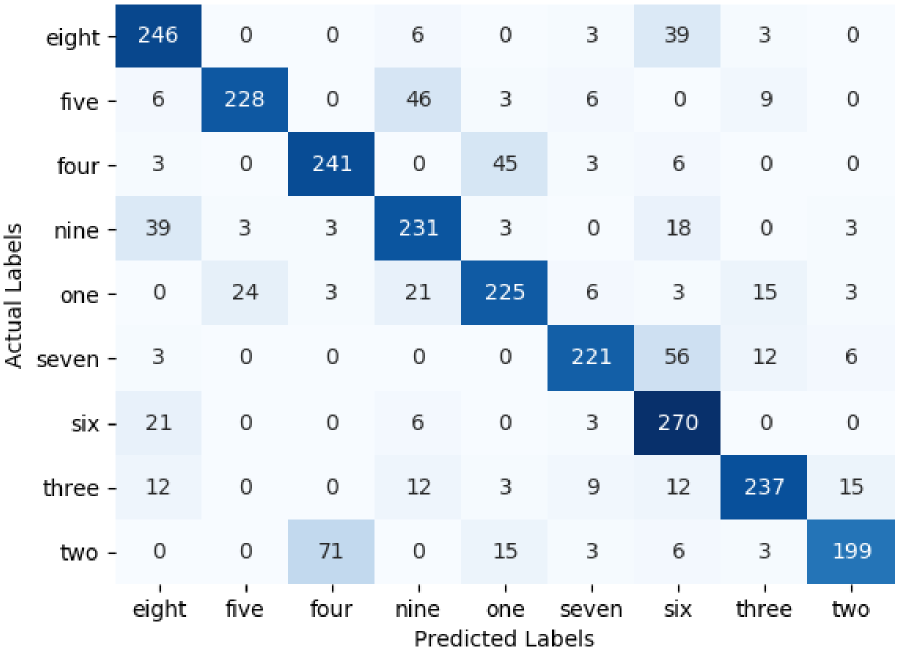


Actual Labels \ Predicted Labels	bin	lay	place	set
bin	640	21	63	36
lay	3	724	9	30
place	6	18	668	54
set	0	42	3	675

Figure 6.5 Commands CNN Confusion Matrix

6.3.2. Numbers CNN

Numbers CNN's architecture was similar to the architecture presented in Figure 6.3 and reached a validation accuracy of ***78.61%*** with confusion matrix as shown in Figure 6.6:

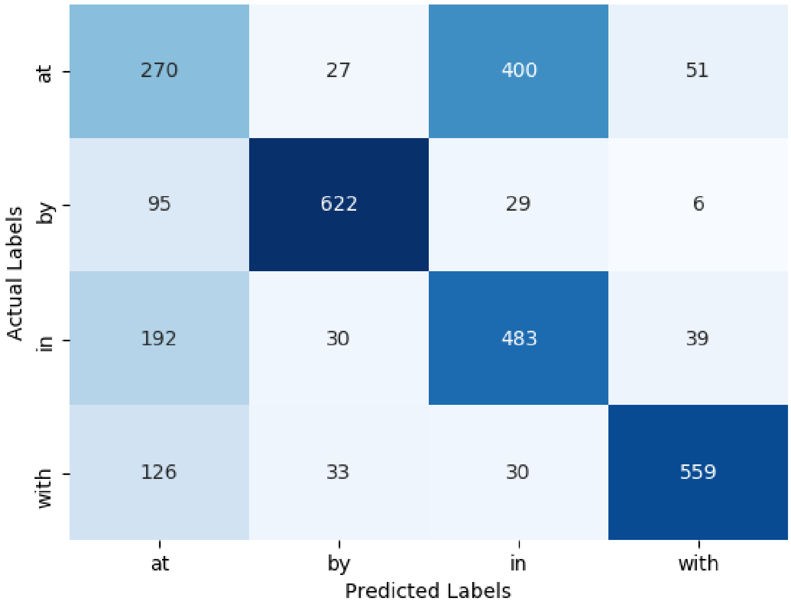


	eight	five	four	nine	one	seven	six	three	two
eight	246	0	0	6	0	3	39	3	0
five	6	228	0	46	3	6	0	9	0
four	3	0	241	0	45	3	6	0	0
nine	39	3	3	231	3	0	18	0	3
one	0	24	3	21	225	6	3	15	3
seven	3	0	0	0	0	221	56	12	6
six	21	0	0	6	0	3	270	0	0
three	12	0	0	12	3	9	12	237	15
two	0	0	71	0	15	3	6	3	199

Figure 6.6 Numbers CNN Confusion Matrix

6.3.3. Prepositions CNN

Prepositions CNN's architecture was similar to the architecture presented in Figure 6.3 and reached a validation accuracy of ***67.48%*** with confusion matrix as shown in Figure 6.7:

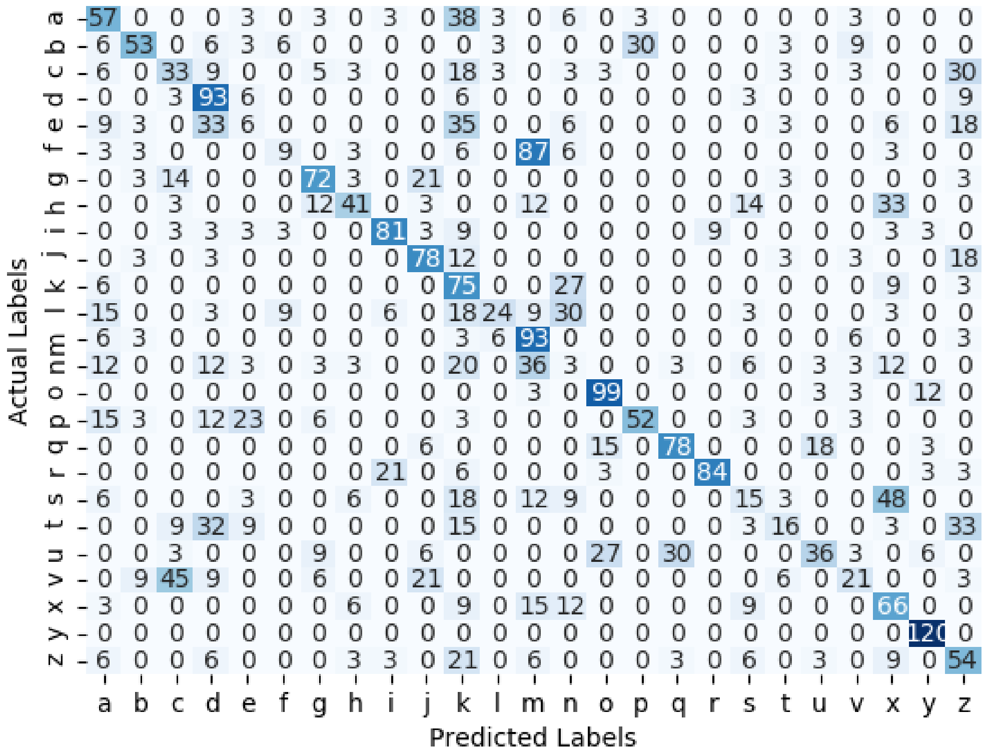


	at	by	in	with
at	270	27	400	51
by	95	622	29	6
in	192	30	483	39
with	126	33	30	559

Figure 6.7 Prepositions CNN Confusion Matrix

6.3.4. Alphabet CNN

The Alphabet CNN had an architecture similar to that presented in Figure 6.3 and reached a validation accuracy of ***48.13%*** with confusion matrix as shown in Figure 6.8:



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	x	y	z
a	57	0	0	0	3	0	3	0	3	0	38	3	0	6	0	3	0	0	0	0	0	3	0	0	0
b	6	53	0	6	3	6	0	0	0	0	0	3	0	0	0	30	0	0	0	3	0	9	0	0	0
c	6	0	33	9	0	0	5	3	0	0	18	3	0	3	3	0	0	0	0	3	0	3	0	0	30
d	0	0	3	93	6	0	0	0	0	0	6	0	0	0	0	0	0	0	3	0	0	0	0	0	9
e	9	3	0	33	6	0	0	0	0	0	35	0	0	6	0	0	0	0	0	3	0	0	6	0	18
f	3	3	0	0	0	9	0	3	0	0	6	0	87	6	0	0	0	0	0	0	0	0	3	0	0
g	0	3	14	0	0	0	72	3	0	21	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3
h	0	0	3	0	0	0	12	41	0	3	0	0	12	0	0	0	0	0	14	0	0	0	33	0	0
i	0	0	3	3	3	3	0	0	81	3	9	0	0	0	0	0	0	9	0	0	0	0	3	3	0
j	0	3	0	3	0	0	0	0	0	78	12	0	0	0	0	0	0	0	0	3	0	3	0	0	18
k	6	0	0	0	0	0	0	0	0	0	75	0	0	27	0	0	0	0	0	0	0	0	9	0	3
l	15	0	0	3	0	9	0	0	6	0	18	24	9	30	0	0	0	0	3	0	0	0	3	0	0
m	6	3	0	0	0	0	0	0	0	0	3	6	93	0	0	0	0	0	0	0	0	6	0	0	3
n	12	0	0	12	3	0	3	3	0	0	20	0	36	3	0	0	3	0	6	0	3	3	12	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	3	0	99	0	0	0	0	0	3	3	0	12	0
p	15	3	0	12	23	0	6	0	0	0	3	0	0	0	0	52	0	0	3	0	0	3	0	0	0
q	0	0	0	0	0	0	0	0	0	6	0	0	0	0	15	0	78	0	0	0	18	0	0	3	0
r	0	0	0	0	0	0	0	0	21	0	6	0	0	0	3	0	0	84	0	0	0	0	0	3	3
s	6	0	0	0	3	0	0	6	0	0	18	0	12	9	0	0	0	0	15	3	0	0	48	0	0
t	0	0	9	32	9	0	0	0	0	0	15	0	0	0	0	0	0	0	3	16	0	0	3	0	33
u	0	0	3	0	0	0	9	0	0	6	0	0	0	0	27	0	30	0	0	0	36	3	0	6	0
v	0	9	45	9	0	0	6	0	0	21	0	0	0	0	0	0	0	0	0	6	0	21	0	0	3
x	3	0	0	0	0	0	0	6	0	0	9	0	15	12	0	0	0	0	9	0	0	0	66	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0
z	6	0	0	6	0	0	0	3	3	0	21	0	6	0	0	0	3	0	6	0	3	0	9	0	54

Figure 6.8 Alphabet CNN Confusion Matrix

The reason for such confusion is that the English alphabet letters don't just depend on the lips' movement, they rely mainly on the ***tongue*** and ***air*** movement [27] as shown in Figure 6.9:

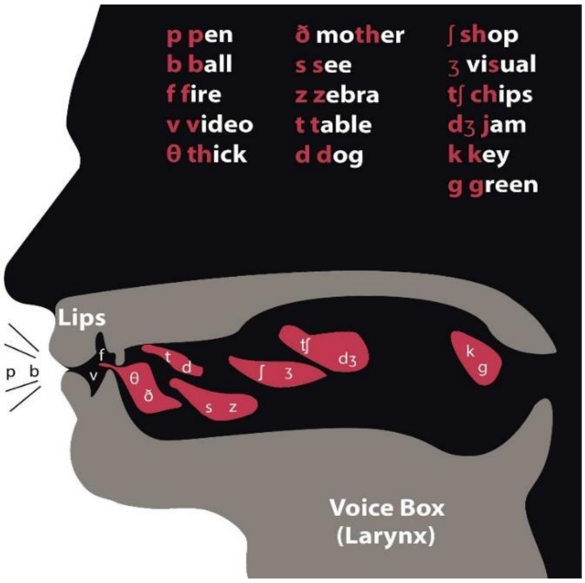


Figure 6.9 English Phonemic Chart

Chapter 7: System Testing and Verification

This chapter discusses the testing and verification of our project and its modules.

7.1. Testing Setup

Inspired by the JUnit testing framework for Java, unit test is a testing framework for Python programs that is already built-in within the Python distribution since Python 2.1.

There are three types of possible test outcomes:

- OK – This means that all the tests are passed.
- FAIL – This means that the test did not pass and an Assertion Error exception is raised.
- ERROR – This means that the test raises an exception other than Assertion Error.

7.2. Testing Plan and Strategy

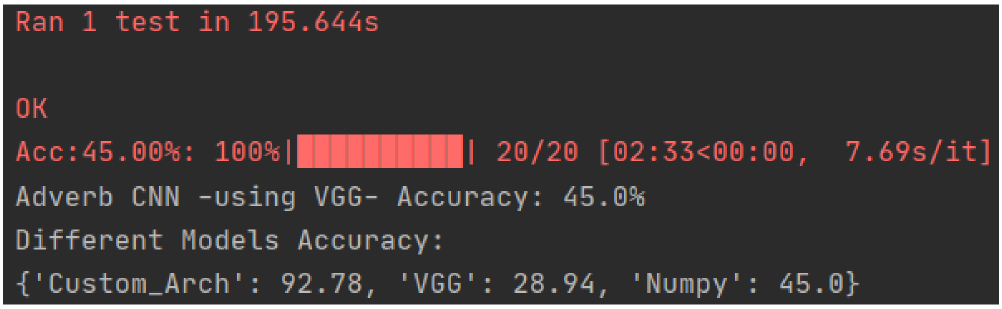
The plan for testing our project was to provide a series of unit tests for the necessary modules to validate that they are working seamlessly and to avoid any crashes/errors that may arise during runtime.

7.2.1. Module Testing

We applied testing to each module on its own at first then started integrating the modules. The applied Tests were on:

➢ **CNN Adverb Accuracy**:

By calling the function that provides the accuracy of each implementation of the Adverb CNN and comparing their results with our pre-known accuracies to be able to detect any malfunction that may occur in the loading of the models or reading of test images.



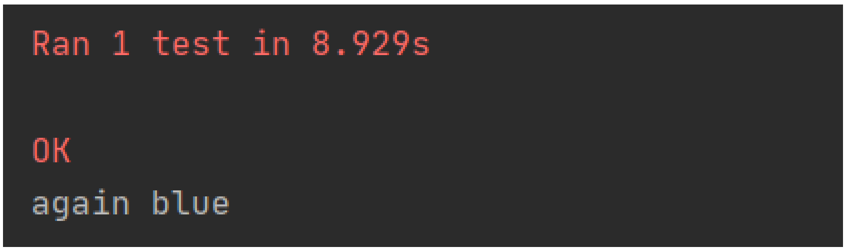
```
Ran 1 test in 195.644s

OK
Acc:45.00%: 100%|██████████| 20/20 [02:33<00:00,  7.69s/it]
Adverb CNN -using VGG- Accuracy: 45.0%
Different Models Accuracy:
{'Custom_Arch': 92.78, 'VGG': 28.94, 'Numpy': 45.0}
```

Figure 7.1 Adverb CNN's Test

- **Project Prototype Output**:

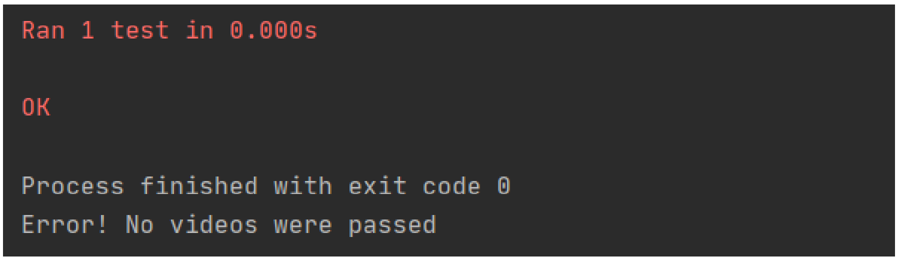
This test verifies the process of receiving a group of videos, manipulating them and passing them to their respective CNNs to predict the labels and provide the output sentence.

If no videos were provided to the module, then we return an error message instead of the output sentence to prevent the app from crashing.



```
Ran 1 test in 8.929s

OK
again blue
```

Figure 7.2 Project Prototype Test



```
Ran 1 test in 0.000s

OK

Process finished with exit code 0
Error! No videos were passed
```

Figure 7.3 Project Prototype No Videos Passed

- **Lips Extraction**:

By counting the numbers of dropped frames that failed to have the lips region extracted from and then dividing them on the total number of frames received from video that provides the accuracy of each implementation of the Lip Extraction and comparing their results with our pre-known accuracies to be able to detect any malfunction that may occur in the loading of the models or reading of test videos, the following graphs shows the average accuracy and time taken to extract lips by each module.

- Modules Accuracy:

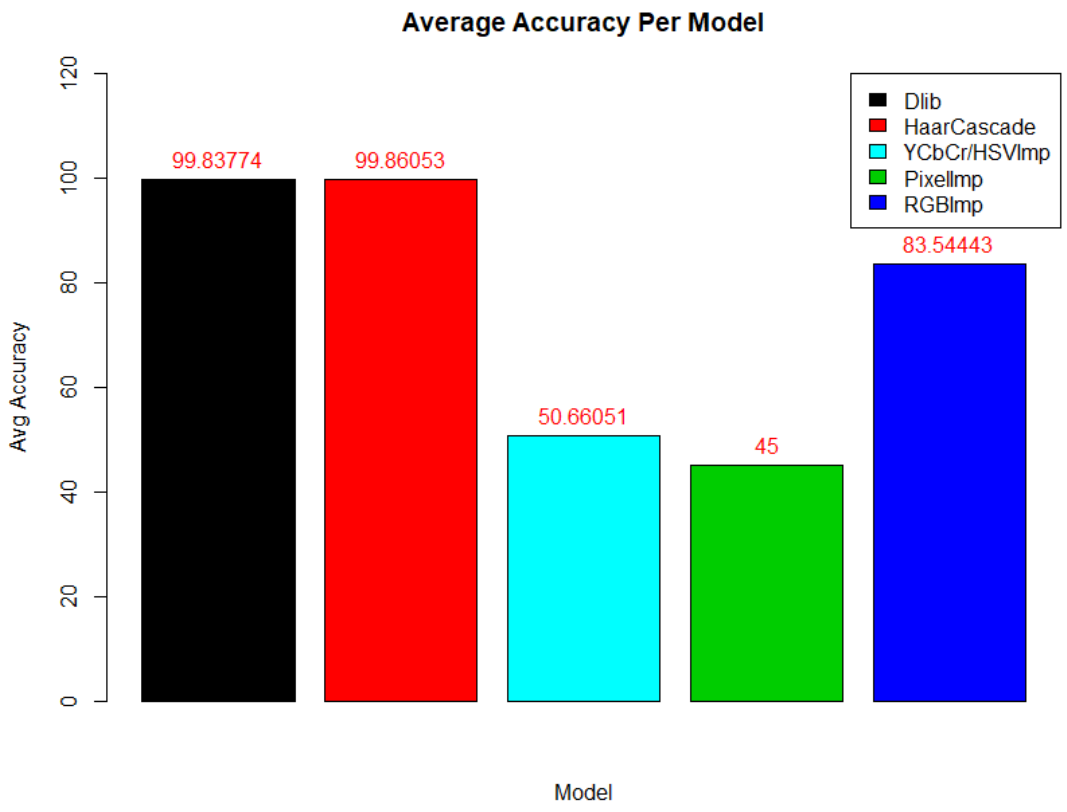


Figure 7.4 Modules Average Accuracy on Test Videos

- Modules Timing:

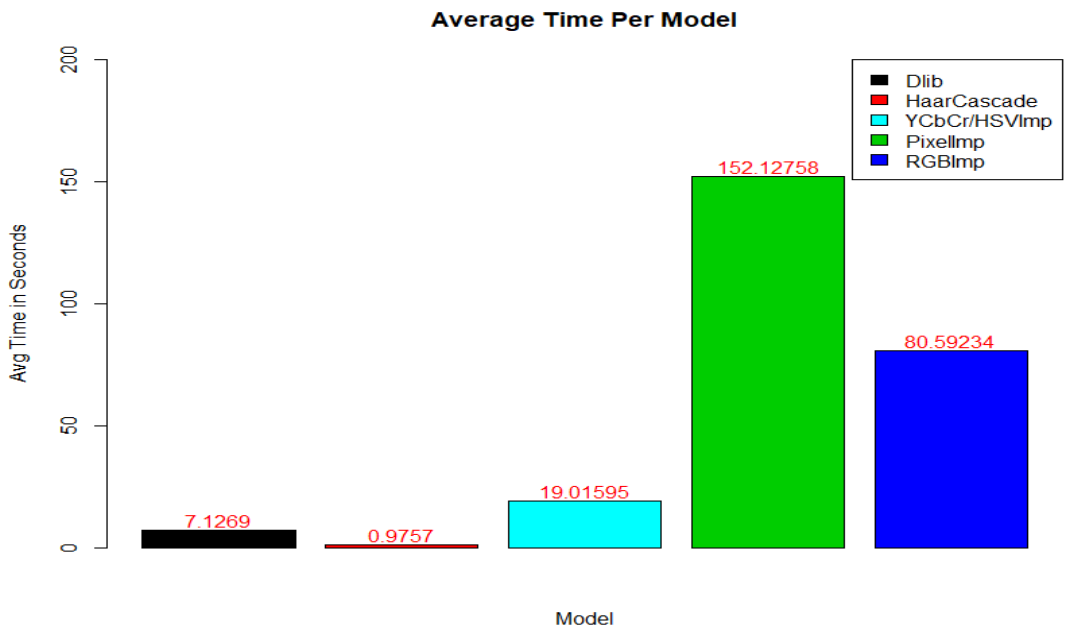


Figure 7.5 Modules Average Timing on Test Video

7.3. Testing Schedule

The testing schedule was divided into 3 categories which are unit testing, integration testing, system testing, those three set of testing were conducted during the development phase of each part.

7.3.1. Unit Testing Schedule:

From the beginning of the development, each unit were independently tested using different methods that fits the model, these tests were conducted each time a specific unit was modified

Test Scenario ID	Image Processing	**Test cases ID**	Image processing-A
Test Cases Description	Mouth ROI extraction	**Test priority**	High
Pre-Requisite	Frames extracted from video	**Post-Requisite**	NA

Test Execution Schedule							
Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
28-Feb-2020	YCbCr/HSV	Run main file	600 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	2-10 images of mouth ROI	Fail	Module fails due to the incorrect segment
29-Feb-2020	YCbCr/HSV	Run main file	600 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	15-25 images of mouth ROI	Pass	Module successes in some cases according to the input frames
5-April-2020	Haar Cascade	Run main file	2000 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	24-25 images of mouth ROI	Pass	Module shows high performance with a high accuracy
6-April-2020	Dlib	Run main file	2000 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	24-25 images of mouth ROI	Pass	Slower than haar cascade but showed less errors
10-May-2020	Pixel Level	Run main file	600 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	18-25 images of mouth ROI	Pass	Module shows same results are YCbCr/HSV

18-May-2020	RGB/RGB Normalized	Run main file	600 Videos of 2 different speakers (25 fps)	25 images of mouth ROI	20-25 images of mouth ROI	Pass	Module showed much better accuracy than YCbCr/HSV

Test Scenario ID	Neural Network	**Test cases ID**	Neural Network-B
Test Cases Description	Extraction Sentence/Word from stream of frames	**Test priority**	High
Pre-Requisite	Mouth ROI Stream	**Post-Requisite**	NA

Test Execution Schedule							
Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
28-Feb-2020	Command CNN	Run main file	Multiple stream of mouth ROI images	Word mentioned in each stream	Words extracted with an accuracy up to 90%	Pass	Model performs well
5-April-2020	Adverb CNN	Run main file	Multiple stream of mouth ROI images	Word mentioned in each stream	Words extracted with an accuracy up to 92%	Pass	Model performs well
	Numbers CCN	Run main file	Multiple stream of mouth ROI images	Word mentioned in each stream	Words extracted with an accuracy up to 78%	Pass	Model Performs well
10-May-2020	Alphabet CNN	Run main file	Multiple stream of mouth ROI images	Word mentioned in each stream	Words extracted with an accuracy up to 42%	Fail	Accuracy Soaring is due that some alphabet depends on tongue not only the lips movement
	Colors CNN	Run main file	Multiple stream of mouth ROI images	Word mentioned in each stream	Words extracted with an accuracy up to 90%	Pass	Model performs well
18-May-2020	Prepositions CNN	Run main file	Multiple stream of mouth	Word mentioned	Words extracted with an	Pass	Model suffers from drop in

			ROI images	in each stream	accuracy up to 67%		accuracy due to using only mouth ROI

Test Scenario ID	Front End	**Test cases ID**	Front End-C
Test Cases Description	Recording Video From mobile camera and receive sentence/word from Server	**Test priority**	High
Pre-Requisite	NA	**Post-Requisite**	Sentence/Word Received from server

Test Execution Schedule

Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
28-Feb-2020	Home Screen	Run main file	-	Home page text correctly showing	Home page text correctly showing	Pass	Page renders correctly
5-April-2020	Sentence Structure Screen	Run main file	-	Category buttons are shown and are disabled on click and passes categories array	Buttons working adding to categories array and disables	Pass	Page renders and functions correctly
10-May-2020	Camera Screen	Run main file	Selected Categories array	Selected categories are shown and camera captures videos	Categories are shown and videos are recorded	Pass	Page renders and functions correctly
18-May-2020	Prediction Result Page	Run main file	-	Shows loading and shows predicted	Waits for server response to show predicated sentence to user	Pass	Page renders and functions correctly

Test Scenario ID	Server	**Test cases ID**	Server-D
Test Cases Description	Receiving user video from front end and sending it the image processing then receiving sentence/word from neural network and send it to the front end	**Test priority**	High
Pre-Requisite	Video from front end	**Post-Requisite**	Sentence/Word from neural network

Test Execution Schedule							
Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
10-May-2020	Receive videos	Run main file	-	Save received videos to 'uploads' folder	Receives videos to uploads folder after clearing it	Pass	Works as expected
18-May-2020	Initialize network and send prediction	Run main file	Videos	Initialize network and send predicted sentence	Network predicts and server sends the prediction and clears	Pass	Works as expected

7.3.2. System Integration Testing Schedule

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

Test Scenario ID	Neural Network with image processing	**Test cases ID**	NI-A
Test Cases Description	• Video of speaker passed to image processing unit to extract mouth ROI stream. • ROI stream passed to neural network to extract sentence/Word	**Test priority**	High
Pre-Requisite	Video from speaker/user	**Post-Requisite**	NA

Test Execution Schedule							
Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
10-May-2020	Haar Cascade with CCN files	Calling predict function in test file that calls both modules functions	600-700 videos of speakers	Words of each speaker	Words predicated by each network with different accuracy	Pass	Successfully integrated both models
18-May-2020	RGB/ Normalized RGB with Handmade Network	Calling predict function in test file that calls both modules functions	100-300 videos of speakers	Words of each speaker	Words predicated by each network with different accuracy	Pass	Successfully integrated both models

Test Scenario ID	Front End with Server	**Test cases ID**	FS-B
Test Cases Description	- Video of speaker passed to image processing unit to extract mouth ROI stream. - ROI stream passed to neural network to extract sentence/Word	**Test priority**	Mid
Pre-Requisite	NA	**Post-Requisite**	Server reply

Test Execution Schedule

Date	Module	Action	Input	Expected output	Actual output	Test results	Test comments
28-Feb-2020	Frontend takes videos and saves them locally	Use app's camera functionality	-	Videos saved locally on the device	Videos are saved	Pass	Works as expected
5-April-2020	Frontend sends videos to server	Run main file	-	Videos sent and saved to uploads folder	Videos sent and saved	Pass	Works as expected
10-May-2020	Initialize network and send prediction	Run main file	Videos	Initialize network and send predicted sentence	Network predicts and server sends the prediction and clears	Pass	Works as expected

7.3.3. System Testing Schedule

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications.

Test Scenario ID	Lipify System	**Test cases ID**	Lipify System-A
Test Cases Description	Recording video using the mobile application and extracting the user's word/sentence from the video and display it on the application	**Test priority**	Mid
Pre-Requisite	NA	**Post-Requisite**	NA

Test Execution Schedule							
Date	**Module**	**Action**	**Input**	**Expected output**	**Actual output**	**Test results**	**Test comments**
10-May-2020	Flutter application, Node.js server, Haar Cascade and CNN network	Open the application and choose category then record the video and send it	Video	The sentence/word from the video	The sentence/word from the video depending on each CNN accuracy	Fail	the front end sends darkened images which cause a high error in predicting word/ sentence
18-May-2020	Flutter application, Node.js server, Haar Cascade and CNN network	Open the application and choose category then record the video and send it	Video	The sentence/word from the video	The sentence/word from the video depending on each CNN accuracy	Fail	The mobile application crashes after one prediction
4-Jun-2020	Web Application, Node.js server, RGB/RGB Normalized and Handmade network	Open the website and choose category then record the video and send it	Video	The sentence/word from the video	The sentence/word from the video depending on each handmade network accuracy	Pass	Model works

10-Jun-2020	Flutter application, Node.js server, Haar Cascade and CNN network	Open the application and choose category then record the video and send it	Video	The sentence/word from the video	The sentence/word from the video depending on each CNN accuracy	Pass	Model works

Chapter 8: Conclusion and Future Work

This chapter concludes our project and discusses possible future work.

8.1. Faced Challenges

It has been a long journey; we faced a lot of challenges and managed to overcome them along the way.

8.1.1. Dataset Problems:

From the beginning, searching for a suitable dataset has been really hard. Almost all research concerned with the topic of ***Lip-Reading*** focuses on identifying words or sentences with **audio-visual** assistance and consequently almost all available datasets didn't focus on the speakers' face as the primary source of information alone, for example The Oxford-BBC Lip Reading in the Wild (LRW) Dataset as shown in Figure 8.1.

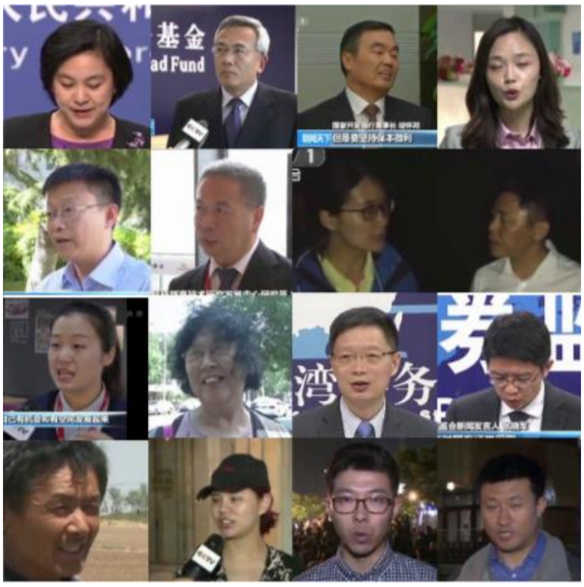


Figure 8.1 LRW Dataset Example

After extensive research we finally found Grid Corpus Dataset, that provides clear view of the speaker's face as shown in Figure 8.2.

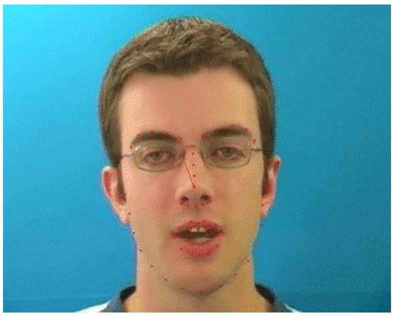


Figure 8.2 Grid Corpus Example

8.1.2. Model Selection:

After we preprocessed our dataset and generated our actual training and validation data, we spent a lot of time researching papers to find a suitable architecture that will be able to predict our labels, the most famous paper amongst them was LipNet[26], But unfortunately the specified CNN in the paper performed poorly with an accuracy of ***11%*** only.

As mentioned in 6.2, we applied our new architectures to the dataset and were able to achieve satisfactory results.

8.2. Gained Experience

This project was one of the biggest – If not the biggest – challenges that faced us since we started our journey towards becoming Computer Engineers. It took a tremendous amount of effort and many sleepless nights, but despite all that, it was one of the most influential experiences from which we emerged enlightened.

8.2.1. Scientific Research

The process of research and planning helped us tremendously in deciding our best courses of action, the areas where we needed to strengthen ourselves and ultimately, we learnt how to actually search papers, summarize them and actually extract relevant data to our project.

After reading a lot of papers, we started to develop an intuition towards analyzing our data and a tolerance towards overcoming obstacles as we started to approach them in methodical ways to determine the adequate course of action to overcome and understand the cause – Like the English Alphabet depending on tongue movement instead of lips movement-.

8.2.2. Software development

Working on this project was a great opportunity for us to apply what we learned throughout the years in an organized manner that would help us maintain the code over long periods of time with proper testing, documentation and organized structure.

We followed -to the best of our abilities- PEP8 and known Python clean code structures to avoid redundancy of code and to write readable functions that are easy to understand to ease the process of development. In addition, we utilized GitHub's marketplace services like ***Code Factor,*** which is an automated code review platform that provided us with actionable insights to increase our code's quality, which now stands at a rating of ***A+***.

8.3. Conclusions

Over the past twenty years, almost every article was dedicated to illustrating the benefit of using visual speech information from a speaker's mouth in addition to the audio signal for the task of speech recognition. Even though all these works have shown that including the visual channel to the speech recognition system greatly improves its performance, no serious attempts were made into creating a sound independent system - like ours or LipNet's -.

In conclusion, the problems facing people with hearing disabilities are numerous and with the current waves of public awareness, we utilized our engineering education to provide them with much needed help by reducing the communication gaps. Of course, the gap is still present as our vocabulary doesn't contain all needed nouns/verbs but It is a monumental stride in a promising direction. Users can currently utter over than ***20 Million*** different possible sentences using our vocabulary – Assuming no repetition of words- as opposed to ***LipNet's 64 Thousand*** predefined sentences.

Our project produced pleasing results in all of its six categories of speech as shown in Figure 8.3:

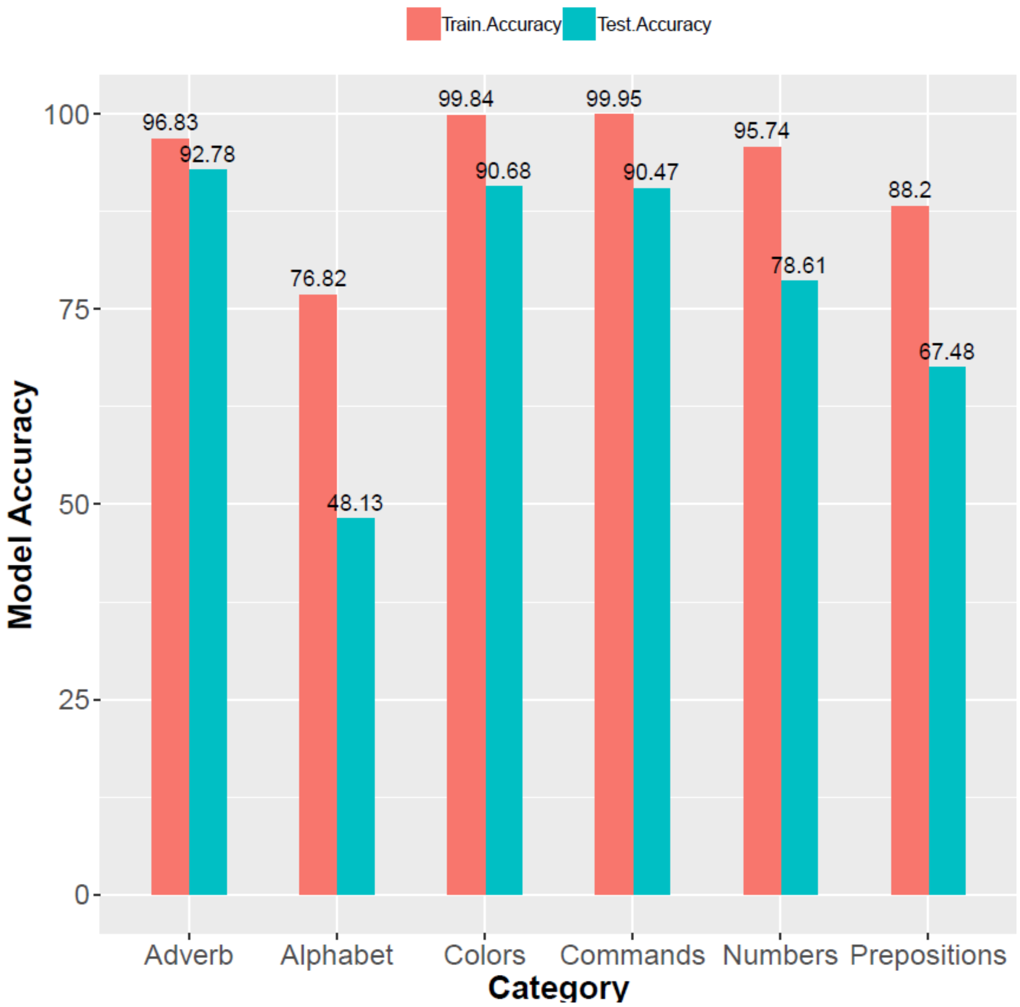


Figure 8.3 Models Accuracy Bar Graph

8.4. Future Work

Probing deeper, the results observed from this project provide a strong foundation for future work in aiding people with hearing disabilities. One area of future work is in combining sign language with mouth movement to compensate for the confusion that occurs from mouth movement alone. Another area we would like to focus on in the future was to explore adding more words to our vocabulary to help diversify speech and give them more degrees of freedom. Our work can be further extended to do sentence level prediction using ***sequence2sequence*** models.

A serious limitation in current audio-visual speech recognition videos is the lack of human pose/viewpoint estimation, if a human's head is tilted in any direction or captured from a side view then it won't be recognized as valid and consequently cannot be identified. In order to overcome these limitations a heavy deal of pre-processing for the videos is needed to accurately ***detect*** and ***predict*** mouth regions if they are obscured in any way.

The applications are endless and are not just limited to the field of hearing-aids, one of the possible uses is that you may be able to look at your phone and mouth a command to your personal assistant app without speaking. Or you might point your phone's camera at someone in a noisy room, and have what they are saying dictated directly into your hearing aids via Bluetooth. More ominously it may offer new surveillance tools that can be used to listen in on distant conversations.

References

[1] British Deaf Association (2015, Sept 7) Fast facts about the Deaf community. Last Accessed 20 March 2020: https://bda.org.uk/fast-facts-about-the-deaf-community/

[2] Liopa. (2019). Sravi: an effective, easy to use communication tool for tracheostomy patients improving patient engagement and autonomy [Online Brochure]. Liopa. Last Accessed 20 March 2020: https://liopa.ai/wp-content/uploads/2019/07/liopa-sravi-voiceless-speech-recognition.pdf

[3] Sprouse, J. (2010) Sign Language [Online slides]. PSYCH 150 / LIN 155 - Psychology of Language. University of California, Irvine. Last Accessed: 4 April 2019: http://www.socsci.uci.edu/˜jsprouse/courses/w13/150/lectures/3.12.sign.language

[4] Hearing for Life (March 2020) The Value of Hearing Services for Vulnerable Australians. Last Accessed: 19 June 2020: https://www.hcia.com.au/hcia-wp/wp-content/uploads/2020/02/Hearing˙for˙Life.pdf

[5] The Canadian Association of the Deaf (2015) Statistics on Deaf Canadians. Last Accessed: 19 June 2020 http://cad.ca/issues-positions/statistics-on-deaf-canadians/

[6] Central Statistics Office, Ireland (2017) Census of Population 2016 – Profile 9 Health, Disability and Carers. Last Accessed: 19 June 2020: https://www.cso.ie/en/releasesandpublications/ep/p-cp9hdc/p8hdc/p9tod/

[7] Mayor of London, Department of Health, London data store (2014) Number of People Registered as Deaf or Hard of Hearing by Age Group, Borough. Last Accessed: 19 June 2020: https://data.london.gov.uk/dataset/number-people-registered-deaf-or-hard-hearing-age-group-borough

[8] National Institute on Deafness and Other Communication Disorders (2016) Quick Statistics About Hearing. Last Accessed: 19 June 2020 https://www.nidcd.nih.gov/health/statistics/quick-statistics-hearing

[9] Mark Lahn (2019) How much does a server cost for a small business. Last Accessed: 19 June 2020: servermania.com

[10] Patel, N. (2003) Lecture 6 - Artificial Neural Networks [Online slides]. 15.062 - Deep Learning. Massachusetts Institute of Technology (MIT). Last Accessed: 10 March 2020: https: https://ocw.mit.edu/courses/sloan-school-of-management/15-062-data-mining-spring-2003/lecture-notes/NeuralNet2002.pdf

[11] Amidi, S. Amidi, A. (2019). Recurrent Neural Networks cheatsheet [Online slides]. CS 230 - Deep Learning. Stanford University. Last Accessed: 10 March 2020: https://stanford.edu/˜shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[12] Saha, A. (2018, Dec 15) A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Last Accessed: 20 March 2020: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[13] Vakhshiteh, Almasganj & Nickabadi (2018). LIP-READING VIA DEEP NEURAL NETWORKS USING HYBRID VISUAL FEATURES. Image Anal Stereo 2018; 36:159-171 doi: 10.5566/ias.1859 Original Research Paper

[14] Extraction of Visual Features for Lipreading IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 2, FEBRUARY 2002.

[15] Sunil S. Morade and Suprava Patnaik "Visual Lip-Reading using 3D-DCT and 3D-DWT and LSDA", International Journal of Computer Applications (0975 – 8887), 2016

[16] Ajith Abraham, Face Detection Using Skin Tone Segmentation. Mumbai, India: 2011

[17] C++ dlib library for face detection explained. (2018, February 13). Davis King: http://blog.dlib.net/

[18] Paul Viola, and Michael Jones "Rapid Object Detection using a Boosted Cascade of Simple Features", Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.

[19] Demedyuk, I. Tsybulskyi, N. (2020) Flutter vs Native Vs React-Native: Examining performance. Last Accessed: 10 May 2020: https://inveritasoft.com/blog/flutter-vs-native-vs-react-native-examining-performance

[20] Naiya Sharma (September 2018) React Native vs. Xamarin vs. Ionic vs. Flutter: Which is best for Cross-Platform Mobile App Development? https://www.apptunix.com/blog/frameworks-cross-platform-mobile-app-development/

[21] Sunil S. Morade & Suprava Patnaik (2014). Lip Reading by Using 3-D Discrete Wavelet Transform with Dmey Wavelet.

[22] Hajraoui, Abdellatif & SABRI, Mohamed. (2014). Face Detection Algorithm based on Skin Detection, Watershed Method and Gabor Filters. International Journal of Computer Applications. 94. 33-39. 10.5120/16349-5695.s

[23] T.F. Cootes and C.J. Taylor and D.H. Cooper and J. Graham (1995). "Active shape models - their training and application". Computer Vision and Image Understanding

[24] E. D. Petajan, Automatic lip-reading to enhance speech recognition, Ph.D. Thesis University of Illinois, 1984.

[25] I. Matthews, G. Potamianos, C. Neti and J. Luettin, "A comparison of model and transform-based visual features for audio-visual LVCSR", IEEE International Conference on Multimedia and Expo, 825–828, 2001.

[26] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson & Nando de Freitas. (2016). LipNet: End-to-End Sentence-level Lipreading. Last Accessed: 8 May 2020: https://arxiv.org/pdf/1611.01599.pdf

[27] English Department (2017) Improve English Pronunciation with the Phonemic Chart. Alba English School, Edinburgh. Last Accessed: 8 May 2020: https://albaenglish.co.uk/blog/improve-english-pronunciation-phonemic-chart

Appendix A: Development Platforms and Tools

This appendix explains used tools, platforms, and hardware kits. Any ready-made module will be mentioned and discussed in this appendix. The appendix is divided into two main sections; one for the hardware and the other is for software.

A.1. Hardware

A description of any used hardware platforms/kit is written in this section.

A.1.1. PC/Laptop

Laptops were used as the main devices for writing, debugging, and maintaining the code and running the server.

A.1.2. Mobile Devices

Mobile devices were used to test the application's usability, speed, and response.

A.2. Software

A description of any used software tool/package is written in this section.

A.2.1. PyCharm

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

A.2.2. Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

A.2.3. Visual Studio Code

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

A.2.4. Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

A.2.5. Notepad++

Notepad++ is a text and source code editor for use with Microsoft Windows. It supports tabbed editing, which allows working with multiple open files in a single window. The project's name comes from the C increment operator. Notepad++ is distributed as free software.

Appendix B: User Guide

Lipify’s user guide on how to install, run, and use the application. The user manual is designed to be as easy and simple and possible to enable all types of users to easily access the application.

After downloading and installing the application, the application will show in your app drawer as shown in Figure B.1.

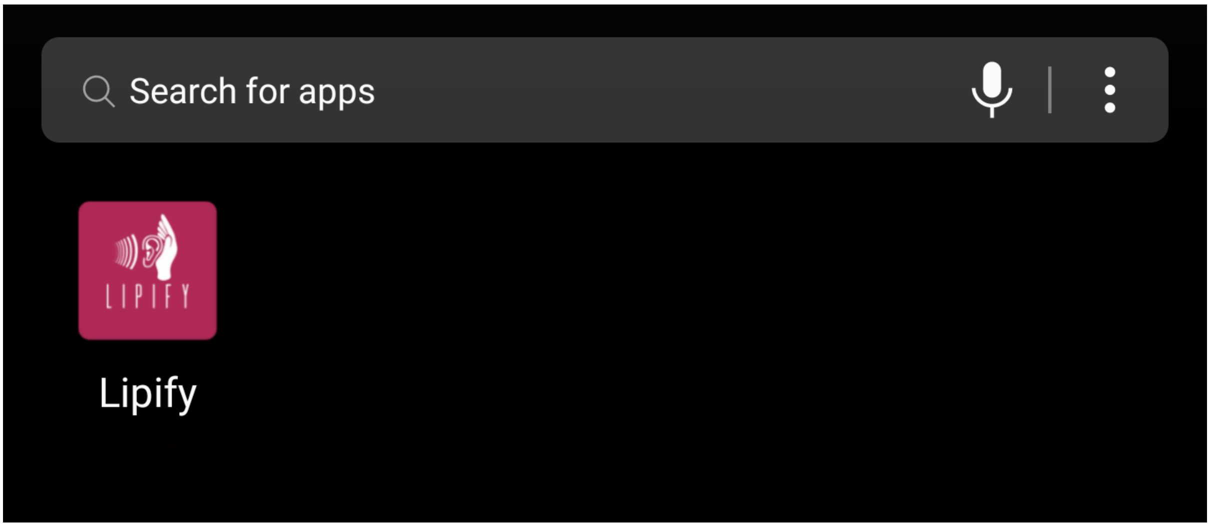
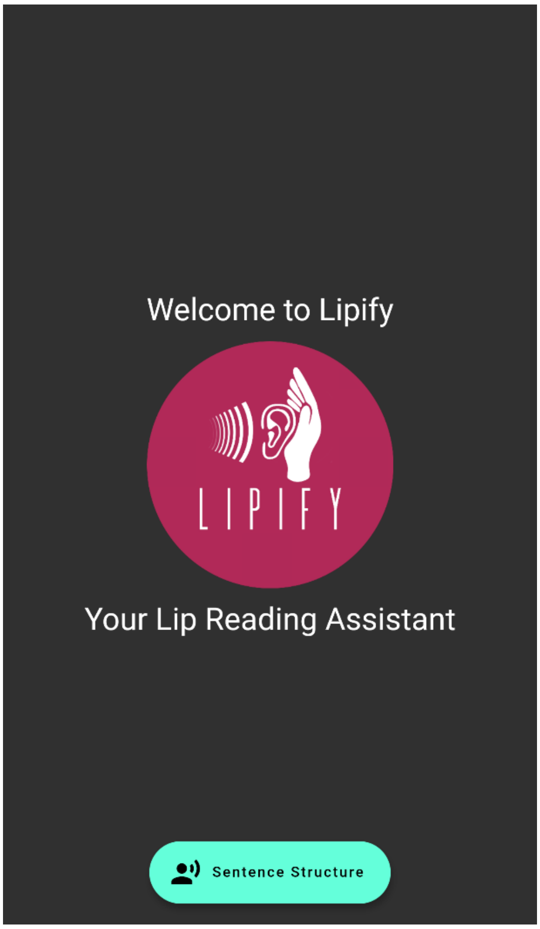


Figure B.1 Lipify in app drawer

Opening the app navigates to the Home Screen shown in Figure B.2. This is just a welcome screen that you just need to click on the “Sentence Structure” button.



Welcome to Lipify

Your Lip Reading Assistant

Figure B.2 Home Screen

B.1. Using Lipify: The Mobile Version

Clicking on that leads to the Sentence Structure screen shown in Figure B.3. The screen shows the categories you can select and a help button on the top right corner.

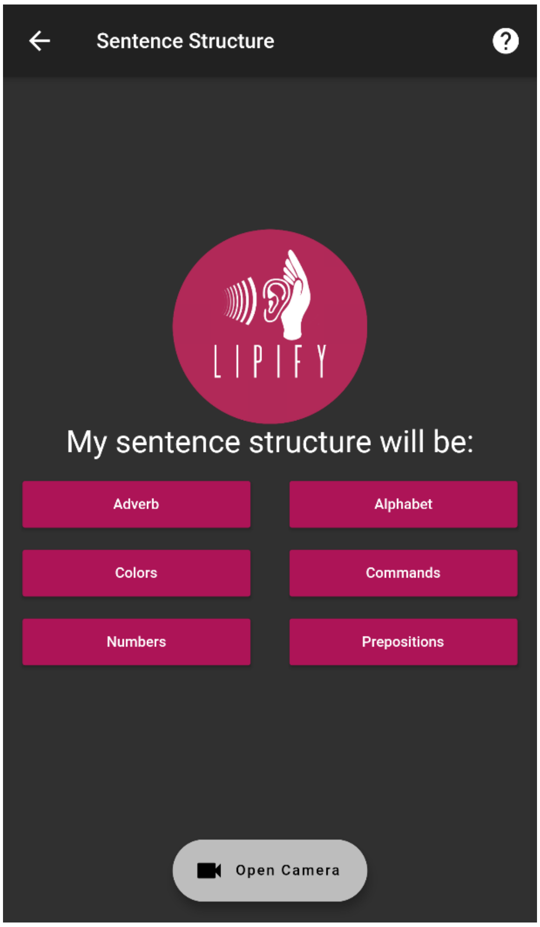
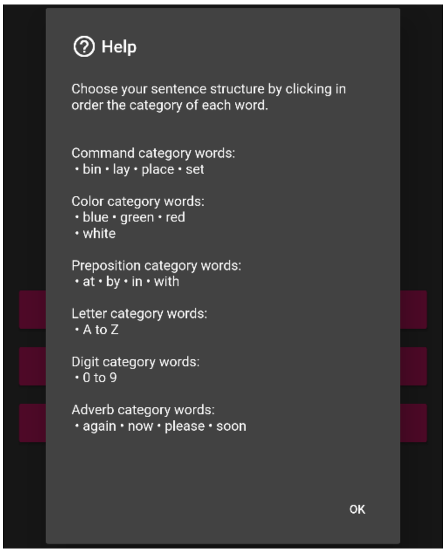


Figure B.3 Sentence Structure Screen

Clicking on the help button shows the Help Dialogue shown in Figure B.4 which explains more about the page.



Help

Choose your sentence structure by clicking in order the category of each word.

Command category words:
• bin • lay • place • set

Color category words:
• blue • green • red
• white

Preposition category words:
• at • by • in • with

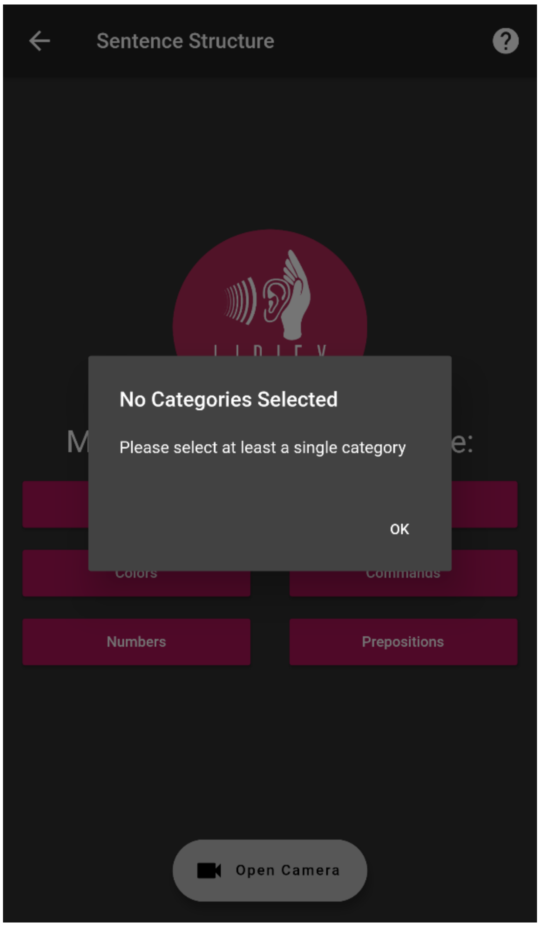
Letter category words:
• A to Z

Digit category words:
• 0 to 9

Adverb category words:
• again • now • please • soon

Figure B.4 Help Dialogue

As you'll notice, the "Open Camera" button is disabled and that is because no categories have been selected yet. So if you click on the button a popup appears shown in Figure B.5.



No Categories Selected

Please select at least a single category

Figure B.5 No Categories Selected Popup

After selecting desired categories in order as shown in Figure B.6 click on the Open Camera button to go to the Camera Screen.

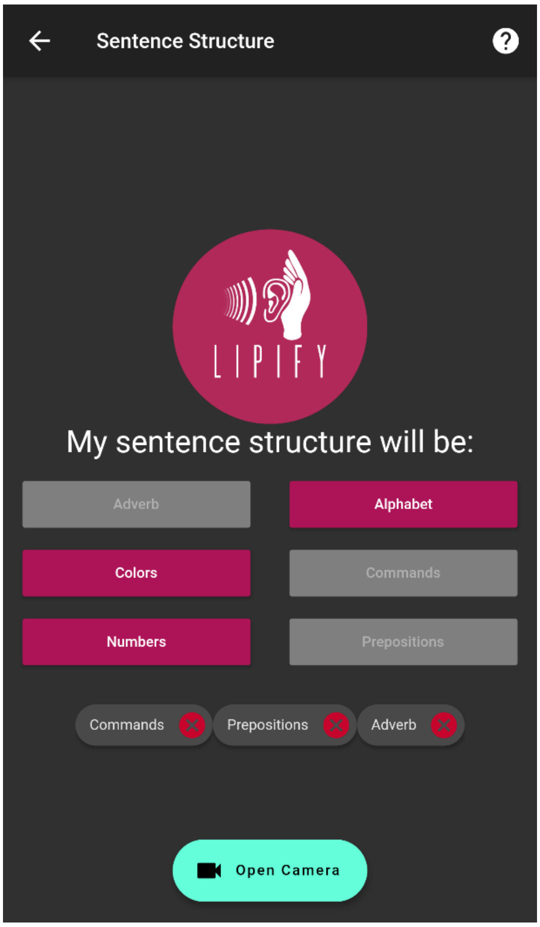


Figure B.6 Selecting categories

The Camera Screen opens allowing you to capture a 2 second video per category by pressing the capture button as shown in Figure B.7.

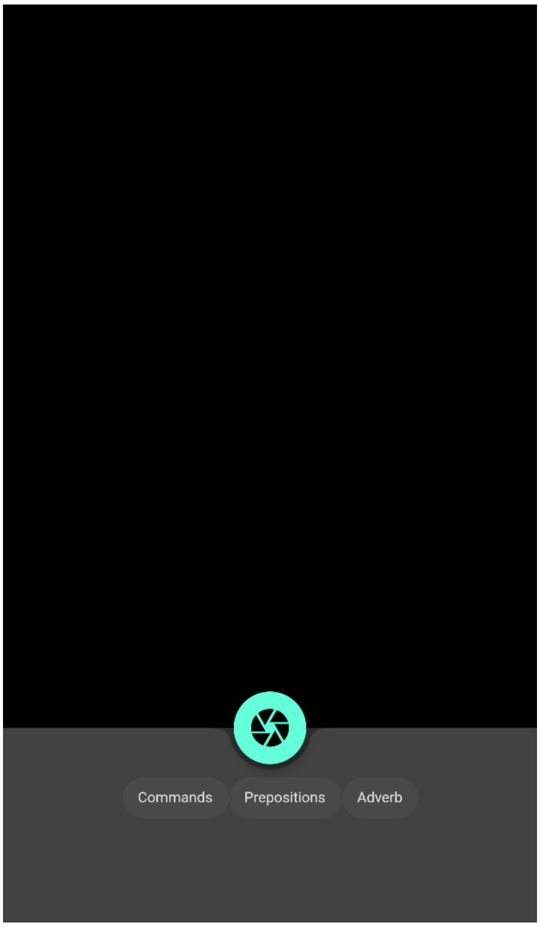


Figure B.7 Camera Screen

After successfully capturing all videos, the application will send the videos to our servers for analysis. So a loading screen appears as shown in Figure B.8.

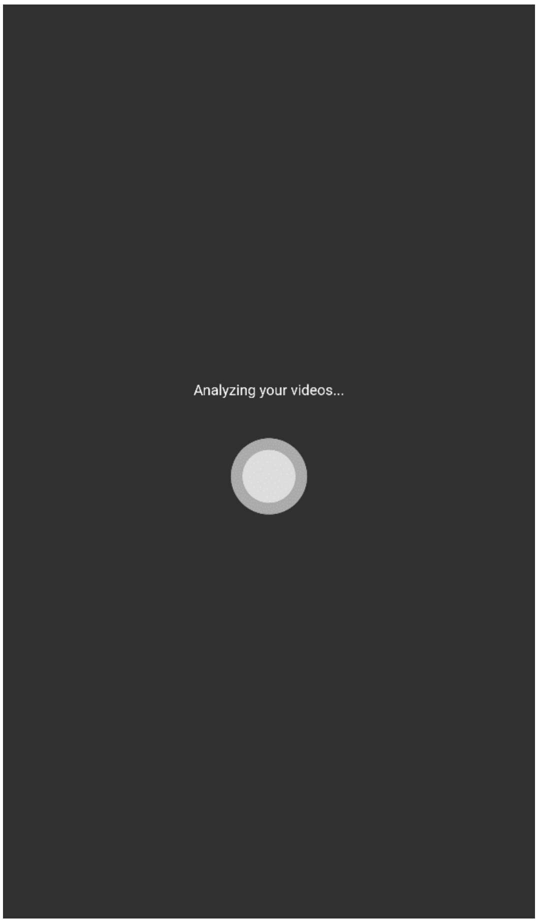
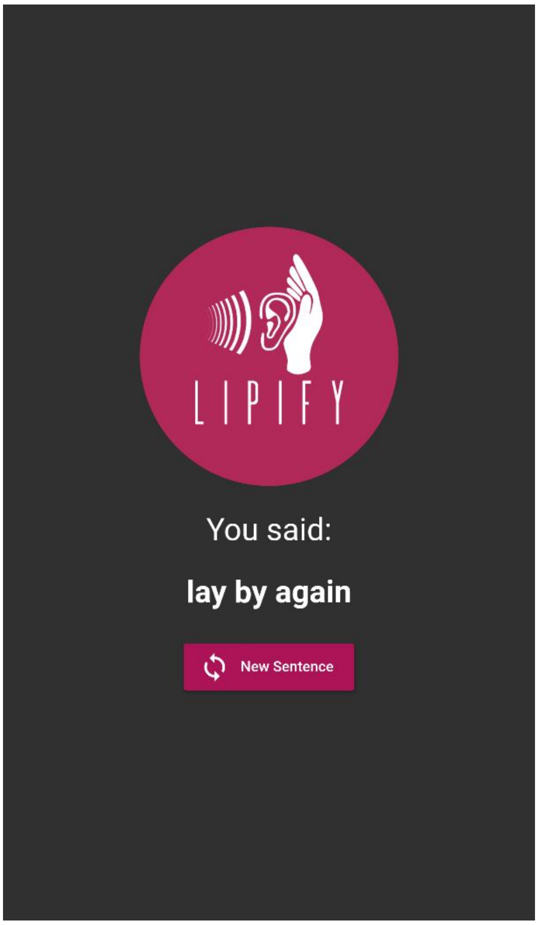

Analyzing your videos...

Figure B.8 Loading Screen

After analyzing your videos, the server responds with the prediction as shown in Figure B.9. The user can then click on the New Sentence button to start all over again with a new sentence.

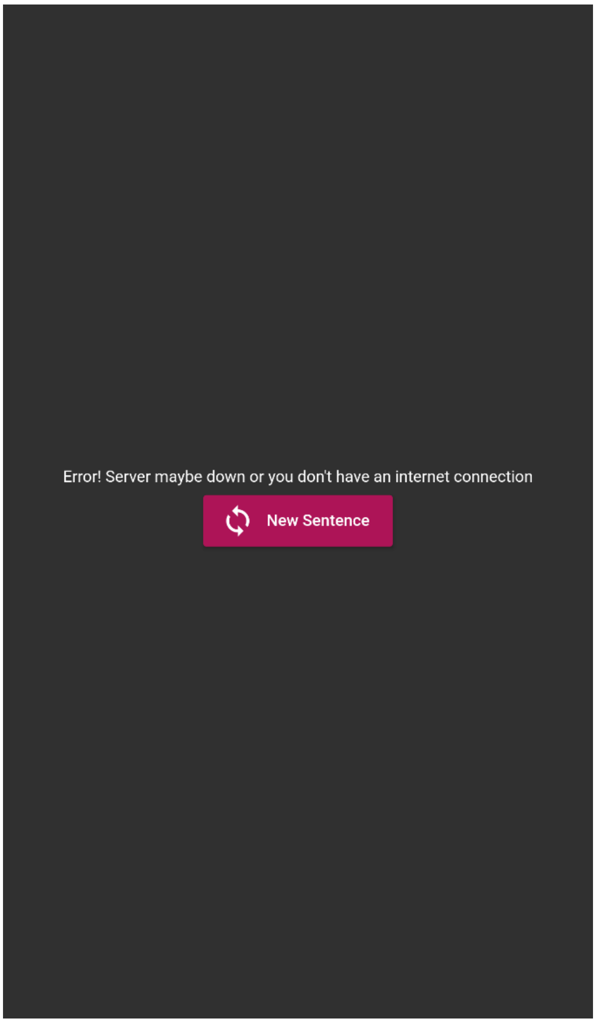


You said:

lay by again

Figure B.9 Prediction Result

If an error arises while connecting to the server, an error is shown to the user along with a new sentence button which repeats the process from the beginning. This is shown in Figure B.10.



Error! Server maybe down or you don't have an internet connection

Figure B.10 Server Error

B.2. Using Lipify: The Website Version

Lipify's website version is similar to the mobile version. The landing/main/home page of the Lipify website is shown in Figure B.11. It is a simple page with Lipify's logo and a button leading to the next page which is the Sentence Structure page.

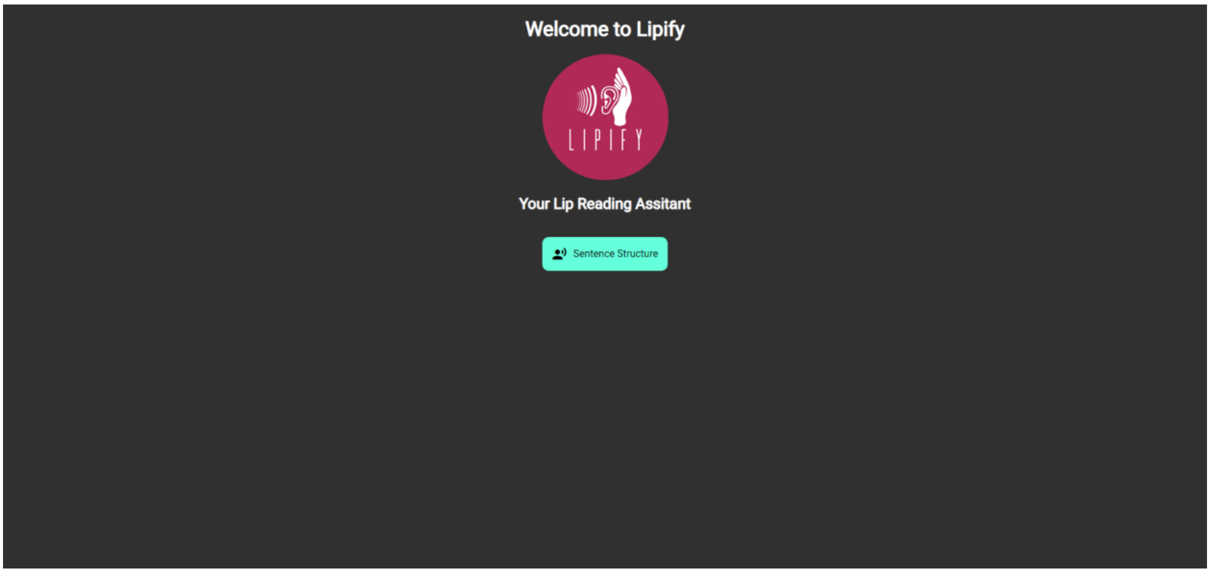


Figure B.11 Website Homepage

The second page is the sentence structure page as shown in Figure B.12. It contains buttons for each category for the user to choose and a button leading to the Camera page.

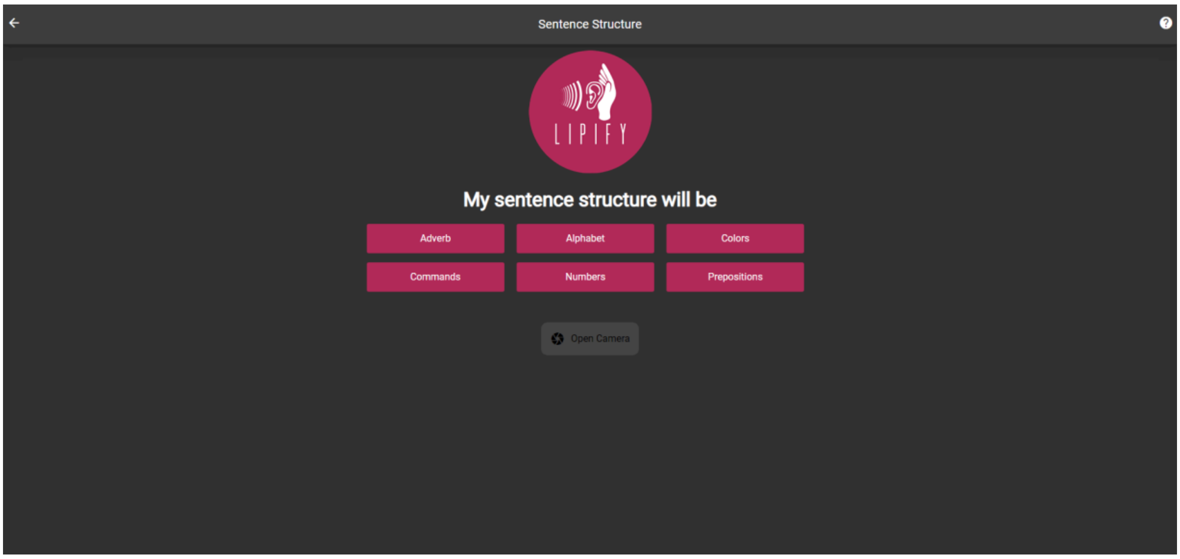
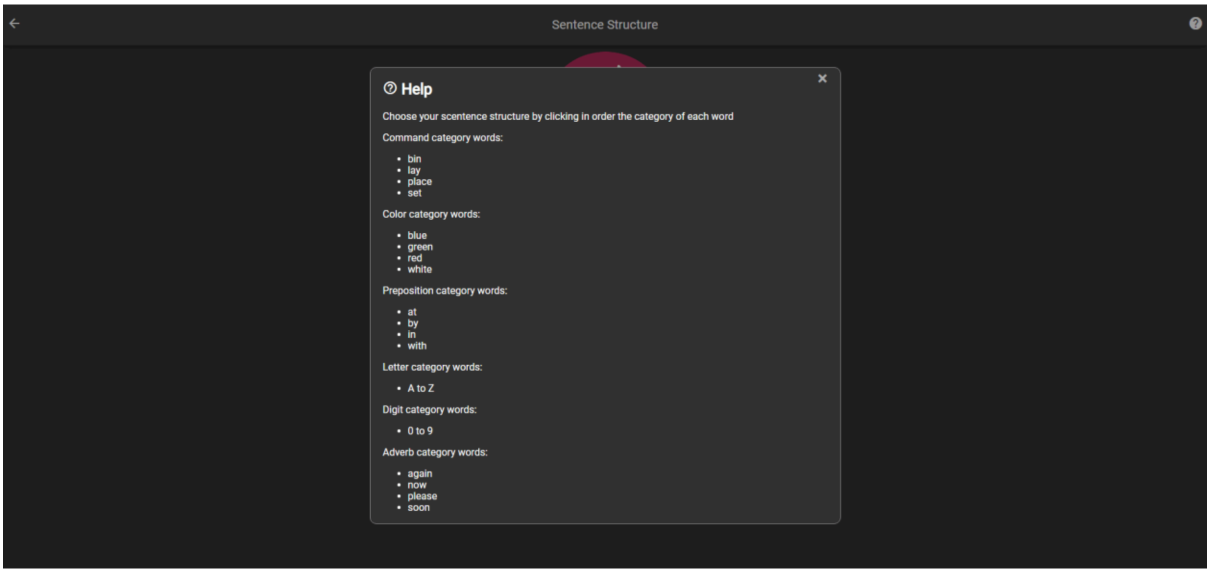


Figure B.12 Sentence Structure Page

A help dialog is shown when the user clicks on the help icon in the navigation bar illustrated in Figure B.13.



Help

Choose your scentence structure by clicking in order the category of each word

Command category words:

- bin
- lay
- place
- set

Color category words:

- blue
- green
- red
- white

Preposition category words:

- at
- by
- in
- with

Letter category words:

- A to Z

Digit category words:

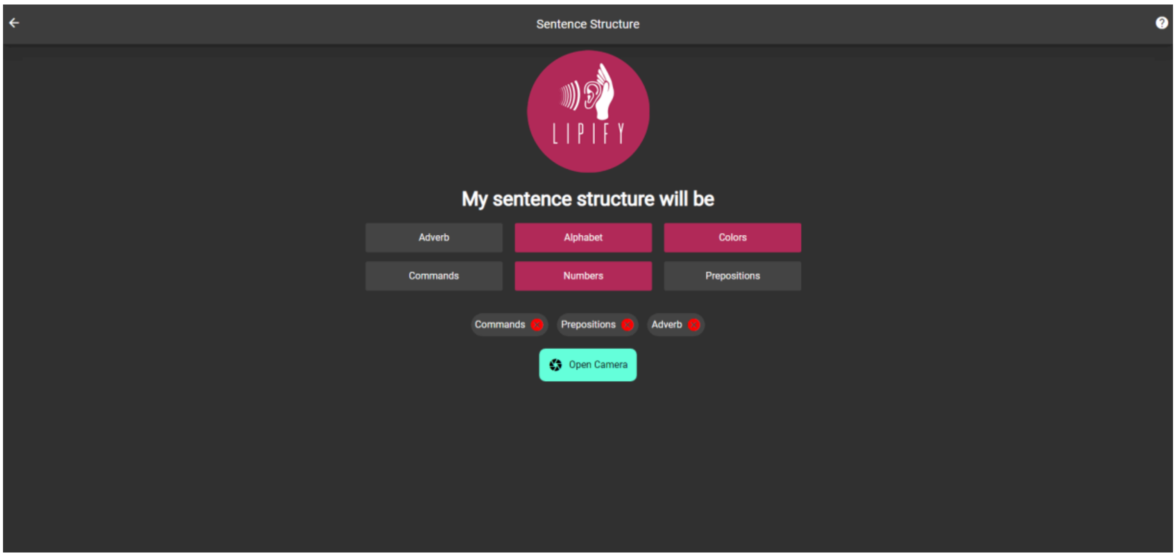
- 0 to 9

Adverb category words:

- again
- now
- please
- soon

Figure B.13 Sentence Structure Help Dialog

As the user selects a category, this category's button is disabled to avoid duplication of categories. The user can also cancel a specific category. This is shown in Figure B.14.



My sentence structure will be

Figure B.14 Selecting Categories

The camera page has a camera live preview and recorded video thumbnail shown in Figure B.15. It shows the user's selected categories and removes them one by one as videos are successfully recorded.

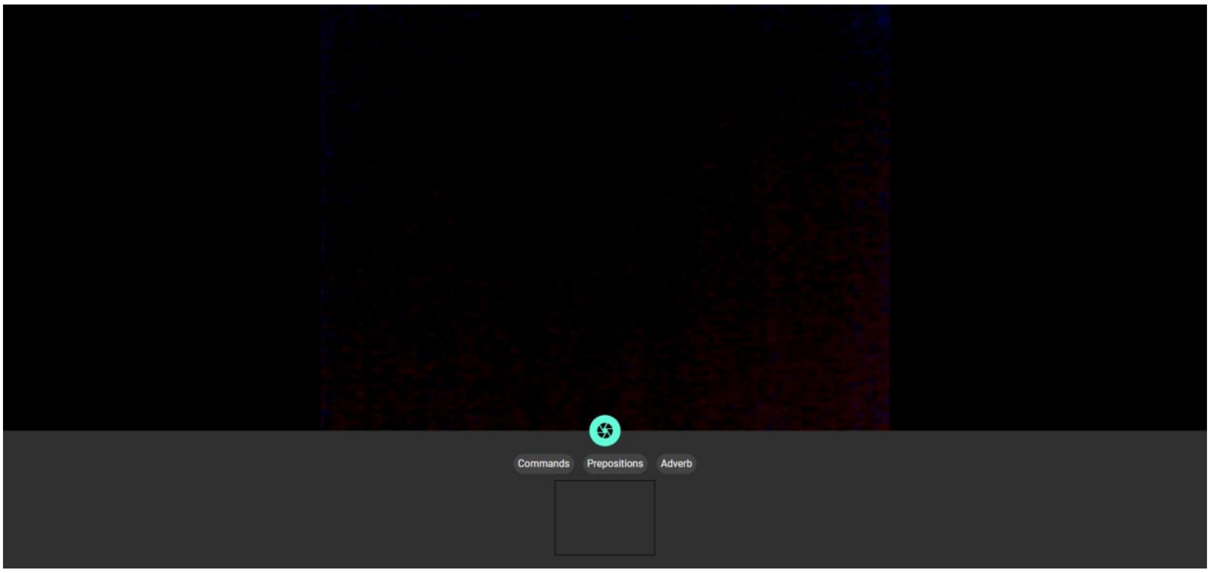


Figure B.15 Camera Page

As a video is being recorded, a red rectangle surrounds the camera preview indicating recording. The camera records for one second where the user can utter the word they desire as per the category. This is clearly shown in Figure B.16.

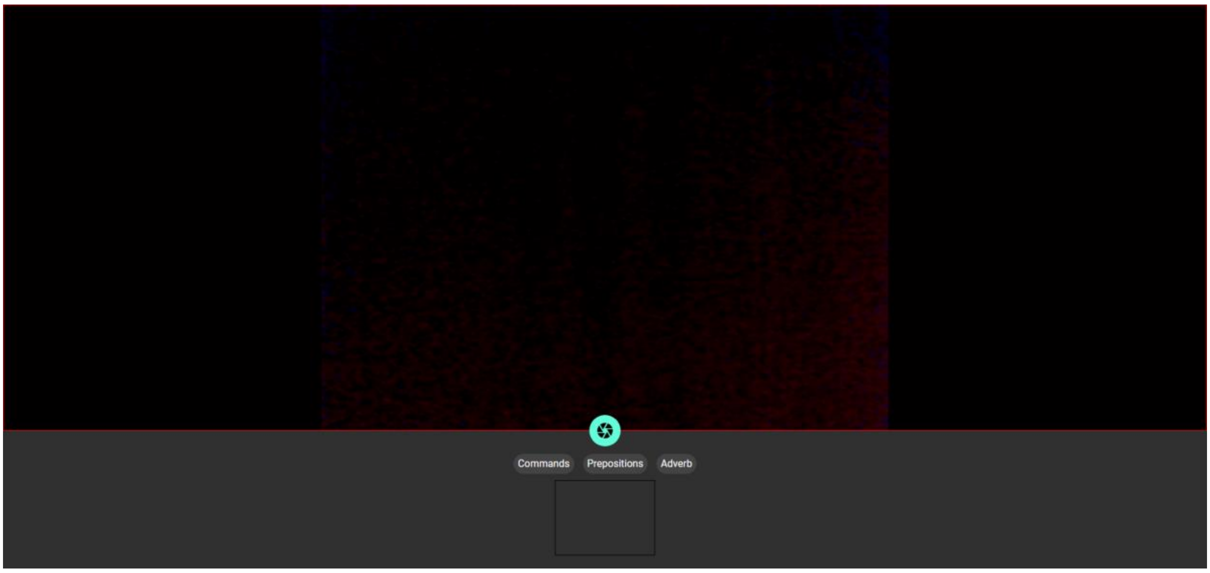
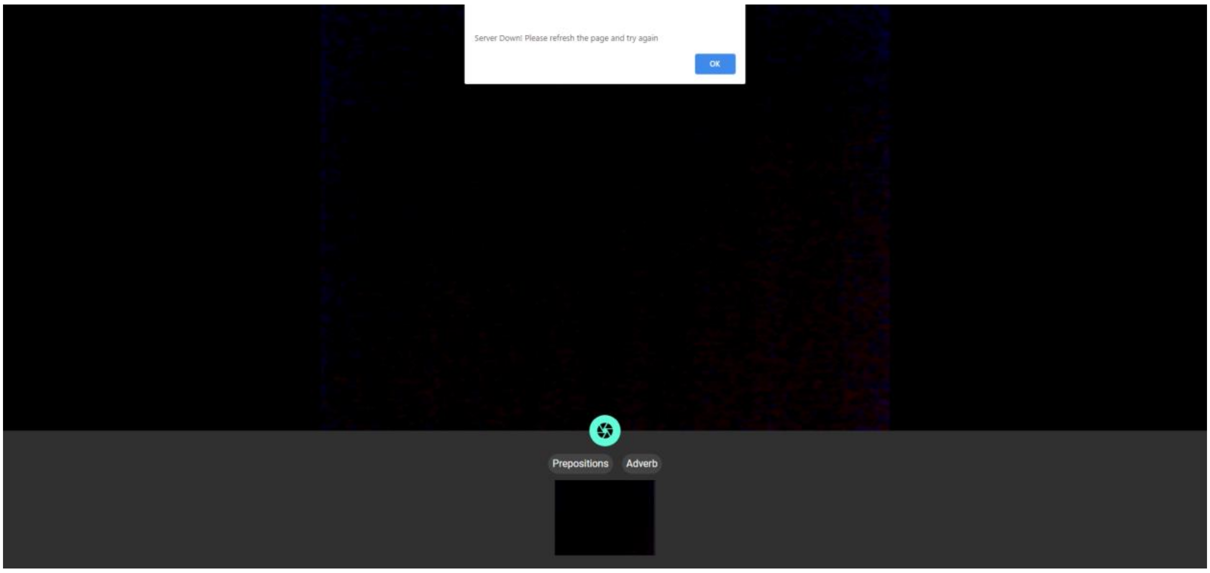


Figure B.16 Camera Page: Recording Video

On successful video recording, a preview of that video is shown beneath the live camera preview as shown in Figure B.17. And if a server error arise, an alert is shown as shown in the same figure.



Server Down! Please refresh the page and try again

Figure B.17 Camera Page: Video Preview

While sending the videos to the server, a loading screen animation is shown to the user as shown in Figure B.18.

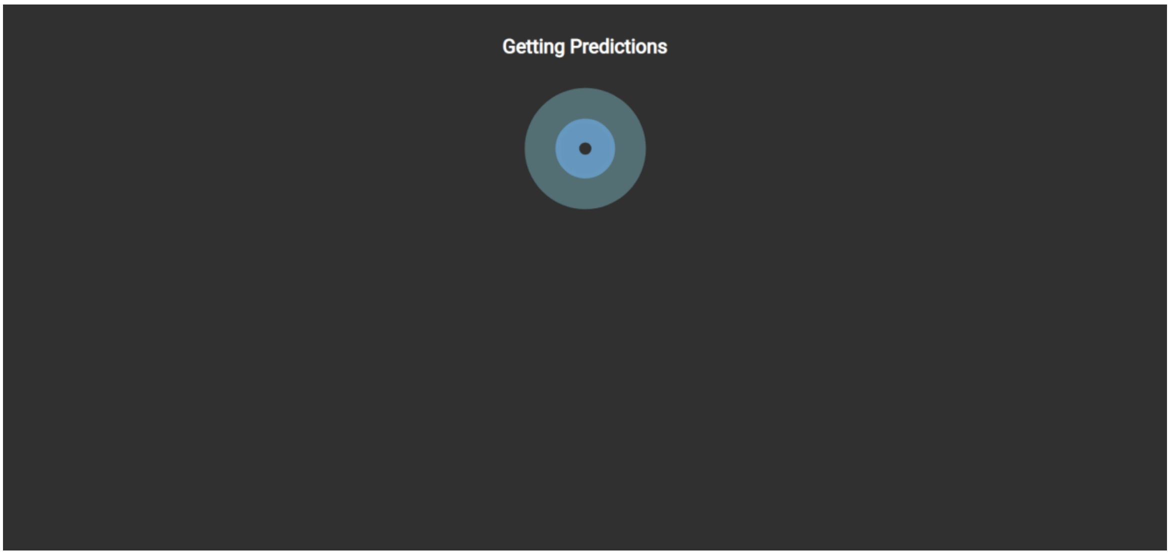
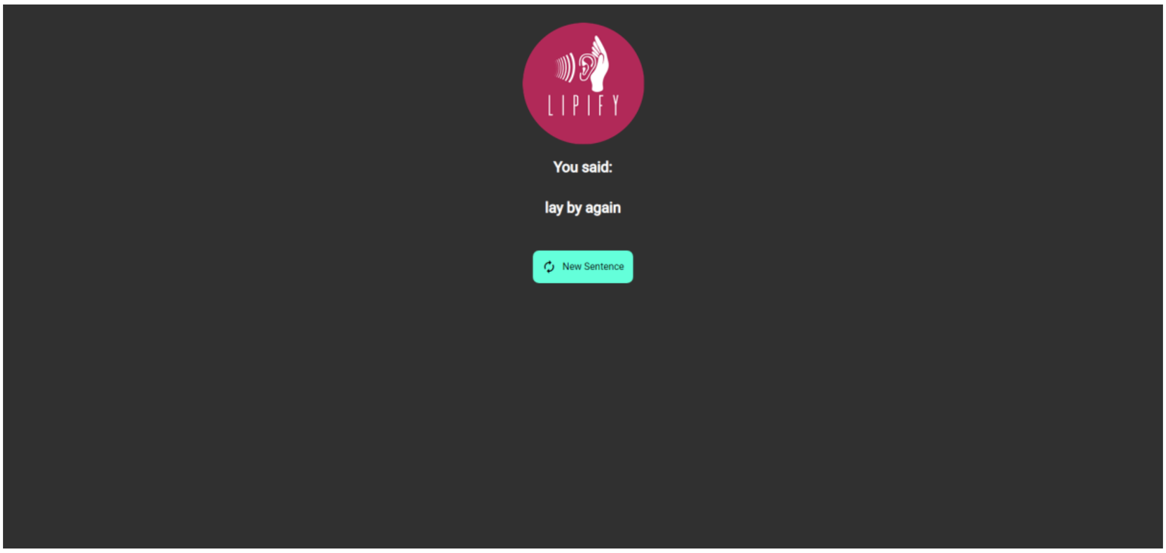


Figure B.18 Prediction Page: Loading

On successful prediction, the server sends back the result and it is shown to the user as shown in Figure B.19.

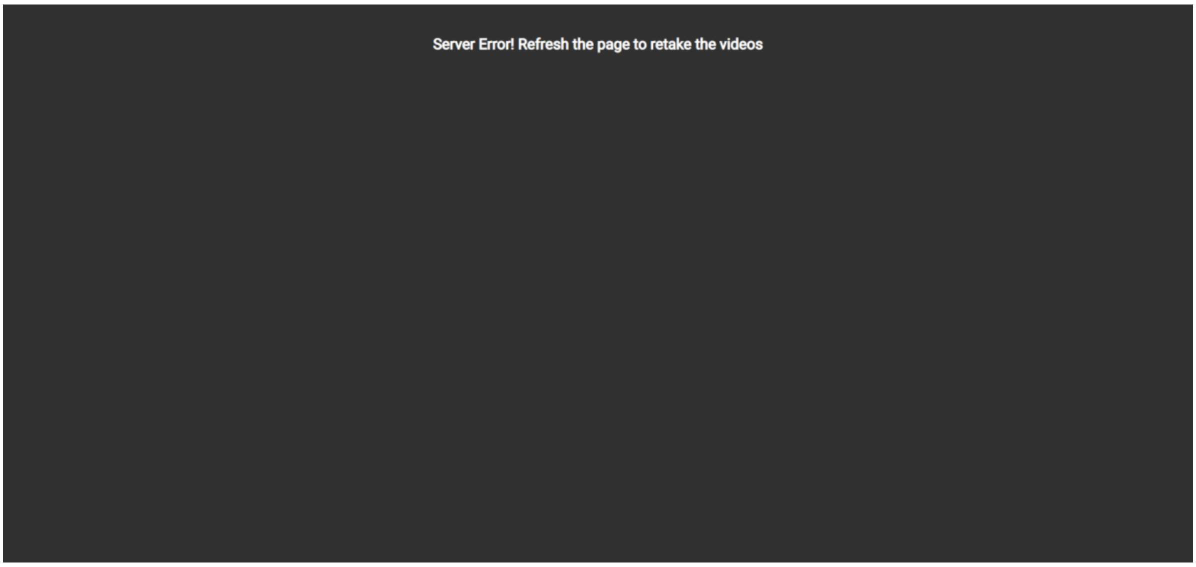


You said:

lay by again

Figure B.19 Server Sent Prediction

If an error happens like the user going offline, the page is updated to indicate the error and offers them to try a new sentence as shown in Figure B.20.



Server Error! Refresh the page to retake the videos

Figure B.20 Prediction Page: Error

The website is also designed to be responsive on all kinds of device screens so that desktop and mobile users will all have the same experience while using Lipify. Figure B.21 shows the website being viewed on an iPhone device.

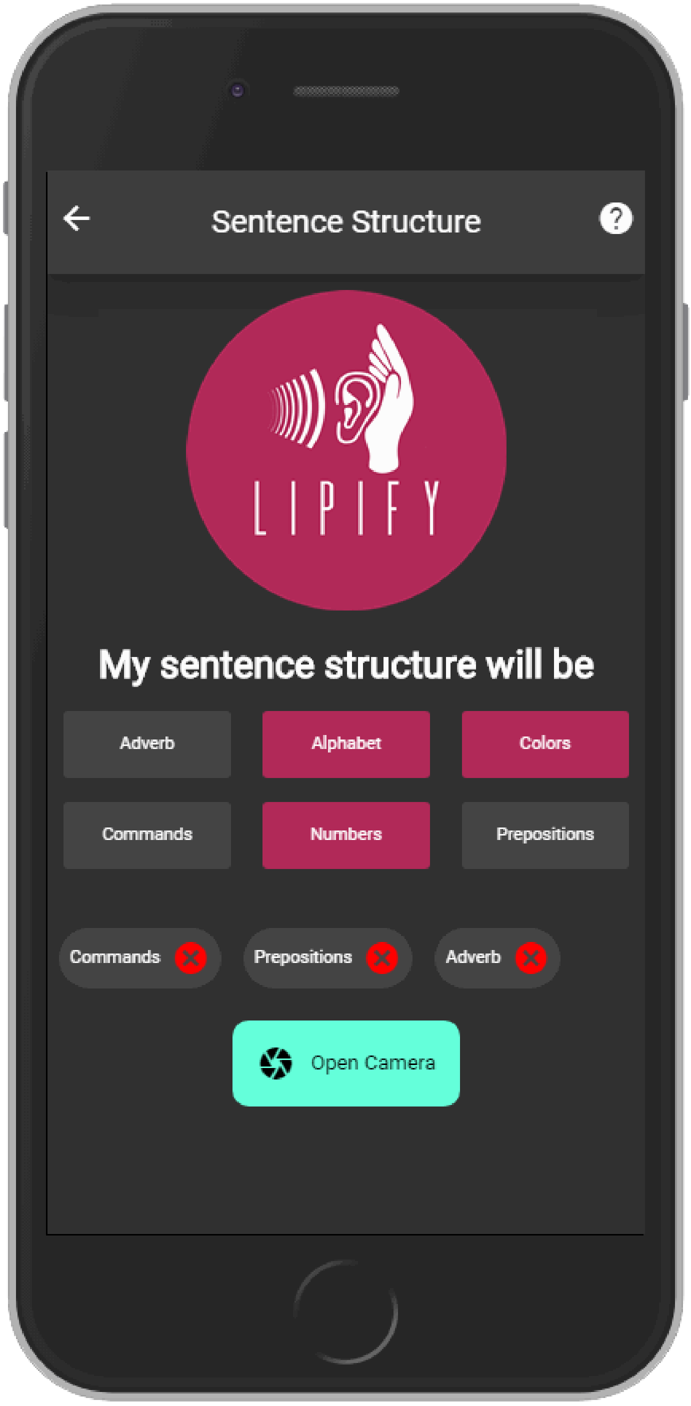


Figure B.21 Website viewed on a mobile

WEE Water Engineering and Environment

STE Structural Engineering

PPC Petro Chemical Engineering

MDE Mechanical Design Engineering

CEM Construction Engineering and Management

CCE Communication and Computer Engineering

AET Architectural Engineering and Technology

Sponsor