

EMOTION DETECTION USING SPEECH AND FACE IN DEEP LEARNING

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Technology degree in
Information Technology

By

Prithiviraj M (Reg No. 39120085)
Shajith Ahamed S (Reg No.39120106)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

MAY 2023

EMOTION DETECTION USING SPEECH AND FACE IN DEEP LEARNING

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Technology degree in
Information Technology

By

Prithiviraj M (Reg No. 39120085)
Shajith Ahamed S (Reg No.39120106)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

MAY 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —AII grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

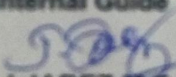
www.sathyabama.ac.in

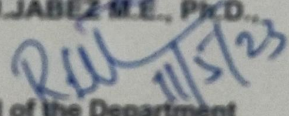
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **PRITHIVIRAJ M (Reg No.39120085)** and **SHAJITH AHAMED S (Reg No. 39120106)** who carried out the Project Phase-2 entitled **"EMOTION DETECTION USING SPEECH AND FACE IN DEEP LEARNING"** under my supervision from January 2023 to May 2023.

Internal Guide

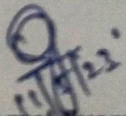

Dr. J. JABEZ M.E., Ph.D.,


Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.,

Submitted for Viva voce Examination held on 11/05/2023

Internal Examiner

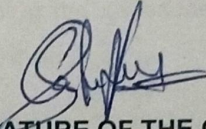

External Examiner

DECLARATION

I **SHAJITH AHAMED S (Reg No. 39120106)** here by declare that the Project Report entitled "**EMOTION DETECTION USING SPEECH AND FACE IN DEEP LEARNING**" done by me under the guidance of **Dr.J.JABEZ M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology

DATE: 11/06/23

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey my thanks to **Dr. T. SASIKALA M.E., Ph. D, Dean**, School of Computing and **Dr. R. SUBHASHINI M.E., Ph.D.**, Head of the Department, Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.J.JABEZ M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

We wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

Humans have a unique ability to demonstrate and understand emotions through a variety of models of communication. Based on their emotions or mood swings we can judge whether the human subject is in good psychological condition or not. The most visible apparent deficiencies of today's Emotion capturing systems were their inability to understand the emotions of such patients like mental health disorder, social emotion Agnosia, alexithymia or even autism by using facial expressions. It can be used in schools to help students who find it difficult to express their feelings (introverts) or who have unstable mental health concerns, such as depression, and hence the teacher's or health workers can communicate with their parents and work through their problems. These days, technology allows employers to recognize individuals who are overly stressed in the workplace and release them from their duties. Emotion detection using speech and face in deep learning has made significant progress in recent years, but there are still several challenges that need to be addressed. Here are some of the main challenges:

Limited Dataset: The availability of labelled datasets for emotion detection is limited, especially for less common emotions or for specific cultural contexts.

Variability in Data: The data used for emotion detection can vary widely in terms of quality, noise, and variability. For example, speech data can be affected by environmental noise, accents, and speaking styles, while facial data can be affected by lighting conditions, facial expressions, and occlusion.

Feature Extraction: Extracting relevant features from speech and facial data can be challenging, especially when dealing with complex emotions that are not easily captured by simple features. This requires careful design of feature extraction algorithms and feature engineering techniques.

Interpretability: Deep learning models are often seen as "black boxes" that are difficult to interpret. This can make it challenging to understand how the model is making decisions and to diagnose errors or biases in the model.

Ethical and Privacy Concerns: Emotion detection using speech and facial data raises ethical and privacy concerns, as it can be used for sensitive applications such as surveillance, emotion profiling, and behavioral prediction. This requires careful consideration of ethical and privacy issues in the design and deployment of deep learning models for emotion detection.

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGS	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 Inferences from literature survey	
	2.2 Open problems in Existing System	6
3	REQUIREMENTS ANALYSIS	7
	3.1 Feasibility Studies/Risk Analysis of the Project	7
	3.2 Hardware Requirements Specification Document	8
	3.3 Software Requirements Specification Document	9
4	DESCRIPTION OF PROPOSED SYSTEM	10
	4.1 Selected Methodology or process model	10
	4.2 Architecture / Overall Design of Proposed System	20
5	IMPLEMENTATION DETAILS	21
	5.1 Testing	21
6	RESULTS AND DISCUSSION	24
7	CONCLUSION	25
	REFERENCES	26
	APPENDIX	
	A. SOURCE CODE	26
	B. SCREENSHOTS	37

Fig NO	List Of Figs	Page No
4.1	Haar Cascade Algorithm	10
4.2	Architecture of Haar Cascade Algorithm	11
4.3	Anaconda Navigator	16
4.4	System Architecture for Speech emotion and recognition	20

CHAPTER 1 INTRODUCTION

Feelings are a crucial part of our day to day lives. As humans, one of the main distinguishing qualities we possess is our ability to demonstrate and understand emotions through a variety of modes of communication. We are able to gauge even the subtlest of emotions in any setting, and these emotions guide our understanding of our interpersonal relationships on a daily basis. As the popularity of speech-based assistants surges, one of the most visibly apparent deficiencies of such systems is their inability to understand the emotions of their users, and further, to demonstrate any kind of emotion in return. Although these systems can perform basic sentiment analysis, they do not inherently process the richness of emotion in everyday speech. Automatic emotion recognition has a natural application in this space (Razuri et al. , 2015), since it can be used not only for automatic user feedback but also to construct more pleasant and natural conversation partners. Emotion recognition technology is essential to such assistants if they are to become more seamlessly integrated into our daily lives. Automatic emotion recognition also has a direct application in the space of medicine and therapy. For those that have social communication disorders like Alexithymia, social-emotional agnosia, or even autism, emotions are very complex to understand and can often feel out of reach. Their inability to detect emotions limits their interactions with familiar people and they are commonly at risk of severely damaging interpersonal relationships. A Scientific American (Serani, 2013) report notes that these diseases are characterized by difficulty identifying different types of feelings, limited understanding of what causes feelings, difficulty expressing feelings, and difficulty recognizing facial cues in others, among others. Tools that can help such individuals identify emotions in people around them could prove to be extremely useful in therapy settings as well as in day-to-day social interactions.

CHAPTER 2

LITERATURE SURVEY

2.1. Inferences from Literature Survey

This section discusses the merits and demerits of the works which are done already related to the proposed work. The author [1] presents a new method for recognizing emotions in conversations using a neural network architecture based on transformers. The research provides an overview of previous work in the field of emotion recognition, which has mainly used traditional machine learning techniques with handcrafted features. Recently, deep learning models like RNNs and CNNs have shown promise. However, these models have limitations in modeling conversational dynamics and long-term dependencies. The proposed approach, CTNet, utilizes the transformer model, which has been successful in natural language processing tasks. CTNet takes a conversation's utterance sequence as input and uses self-attention to model the relationships between them. The resulting representation is then fed through a multi-layer perceptron for emotion classification. The authors evaluate CTNet on two datasets: the Friends TV show corpus and the IEMOCAP dataset of acted conversations. They compare the results of CTNet to several baseline models, including RNNs and CNNs, and find that CTNet outperforms them all. Ablation studies are also conducted to analyse the contribution of different components of the architecture. In conclusion, the article presents a novel approach for recognizing emotions in conversations and demonstrates the effectiveness of the CTNet architecture. The research [2] proposes a new method for speech emotion recognition using a neural network architecture combining CNNs, BLSTMs, attention mechanisms, and skip connections. The authors conduct a literature review of previous work in the field of speech emotion recognition and identify the challenges associated with this task. The proposed approach, ASCB-LSTM, achieves state-of-the-art results on two benchmark datasets, outperforming traditional machine learning algorithms and deep learning models without attention and skip connections. Overall, the research presents a promising solution to speech emotion recognition that combines various neural network architectures to improve accuracy in recognizing emotions from

speech signals. A novel approach using a transformer-based architecture that integrates multiple modalities for continuous affect recognition [3] has been suggested. The research highlights the challenges in previous work that have focused on recognizing emotions from separate modalities. The proposed MEME-T model achieves state-of-the-art results on two benchmark datasets, outperforming traditional machine learning algorithms and deep learning models without multi-modal attention mechanisms. The results suggest that MEME-T is effective in capturing the dynamic relationships between different modalities and can improve the accuracy of affect recognition. This article[4] presents a novel approach to speech emotion recognition using a multimodal neural network architecture. The authors provide an overview of previous work in the field of speech emotion recognition and identify the limitations of traditional methods that rely solely on audio features. They propose a multimodal approach that combines audio, visual, and textual features using a multi-level multi-head fusion attention-based recurrent neural network (MLMHF-Att-RNN).The MLMHF-Att-RNN model selectively weights the importance of different modalities using attention mechanisms and combines them using a fusion approach that integrates multiple levels and heads. The authors evaluate the model on two benchmark datasets, and the results show that the proposed approach outperforms other state-of-the-art models, achieving significant improvements in accuracy. Overall, the research presents a promising approach to speech emotion recognition that leverages multiple modalities using a novel neural network architecture. The results demonstrate the potential of multimodal approaches in improving the accuracy of speech emotion recognition and highlight the importance of considering multiple modalities in this task. A fusion model that combines textual, audio, and visual features to recognize emotions in multimodal data [5] has been suggested. The model is based on BERT and is designed to handle input data of different types and from different sources. To evaluate the proposed model, the authors compare its performance with several state-of-the-art emotion recognition models on two benchmark datasets. The results show that the proposed model performs better in terms of accuracy and F1-score. The research's contributions are significant as multimodal emotion recognition is a vital research area with applications in human-computer interaction, affective computing, and mental

health monitoring. Overall, the research presents a novel approach that leverages the strengths of BERT and addresses the challenge of fusing heterogeneous features.

The author [6] provides an extensive survey of the field of multimodal machine learning, which involves combining information from multiple modalities, such as vision, speech, and text, to solve complex tasks. The authors present a comprehensive taxonomy of multimodal machine learning approaches, which includes feature fusion, decision fusion, and deep fusion. They also discuss various applications of multimodal machine learning, including emotion recognition, action recognition, and speech recognition. The authors compare and contrast different multimodal machine learning techniques, highlighting their strengths and limitations. They also identify open research challenges in the field, such as handling missing modalities, dealing with noisy data, and developing scalable algorithms. Overall, the research provides a valuable resource for researchers interested in multimodal machine learning. The comprehensive survey and taxonomy enable readers to gain a deeper understanding of the field's landscape, identify key challenges, and explore potential solutions. The research's contributions are significant, as multimodal machine learning is an important research area with numerous applications, including human-robot interaction, natural language processing, and multimedia analysis. The research proposes a model for recognizing emotions from speech data, which utilizes Long Short-Term Memory (LSTM) networks with an attention mechanism [7] has been proposed. The model is designed to process short speech segments and identify the underlying emotion. To evaluate the proposed model's performance, the authors conduct experiments on the Emo-DB dataset, which contains recordings of acted emotional speech. The results demonstrate that the proposed model outperforms other state-of-the-art models in terms of accuracy and F1-score. The research's contributions are significant as speech emotion recognition is a crucial research area with a wide range of applications, including affective computing, speech-based human-computer interaction, and mental health monitoring. The proposed model's capability to handle short speech segments and its superior performance make it a promising approach for real-world applications. Overall, the research presents a novel approach to speech emotion recognition that

leverages the strengths of LSTM networks and attention mechanisms. The experimental results validate the effectiveness of the proposed model and highlight its potential for practical applications. The research [8] proposes a method for speech emotion recognition that integrates Hidden Markov Models (HMMs) with a deep neural network (DNN) to enhance its performance. To evaluate the proposed method's effectiveness, the authors conduct experiments on the Interactive Emotional Dyadic Motion Capture (IEMOCAP) dataset, which contains acted emotional speech recordings. The experimental results show that the proposed method outperforms other state-of-the-art models in terms of accuracy, F1-score, and kappa coefficient. The research's contributions are significant as speech emotion recognition is a crucial research area with numerous applications, including affective computing, speech-based human-computer interaction, and mental health monitoring. The proposed method's capability to combine the strengths of traditional machine learning approaches with the power of deep learning makes it a promising approach for real-world applications. Overall, the research presents a novel approach to speech emotion recognition that revisits the traditional HMMs and combines them with a DNN. The experimental results validate the effectiveness of the proposed method and highlight its potential for practical applications. The author proposes a fusion-based approach for recognizing emotions from speech data, which combines both acoustic and text-based features. The authors explore several fusion methods [9], including early, late, and hybrid fusion, to evaluate the proposed approach's effectiveness. To evaluate the proposed approach's performance, the authors conduct experiments on two publicly available datasets, the Interactive Emotional Dyadic Motion Capture (IEMOCAP) and the Friends TV Show dataset. The experimental results show that the proposed approach outperforms other state-of-the-art models in terms of accuracy, F1-score, and kappa coefficient. The research's contributions are significant as recognizing emotions from speech is a crucial research area with a wide range of applications, including affective computing, speech-based human-computer interaction, and mental health monitoring. The proposed fusion-based approach's ability to combine both acoustic and text-based features makes it a promising approach for real-world applications. Overall, the research presents a novel approach to recognizing emotions from speech data

that utilizes fusion-based methods to combine acoustic and text-based features. The experimental results validate the effectiveness of the proposed approach and highlight its potential for practical applications. This author [10] outlines an approach for generalised labelled multi-bernoulli filter-based robust pitch estimation and speaker tracking (GLMB). The proposed method outperforms other state-of-the-art models in terms of accuracy and tracking. The research's contributions are significant for speech processing applications, and the proposed method's potential for practical applications is highlighted.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system is Artificial Neural Network – Multilayer Perceptron (MLP) Artificial Neural Network is equipped for learning any nonlinear function. Henceforth, these organizations are prominently known as Universal Function Approximators. ANNs have the ability to learn loads that map any contribution to the yield. One of the principal purposes for general guess is the enactment work. Enactment capacities acquaint nonlinear properties with the organization. This enables the organization to gain proficiency with any unpredictable connection among input and output . The Expression MLP is applied equivocally allude to a feedforward artificial neural network.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

3.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2. Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. A feasibility study evaluates the project's potential for success.

3.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2. HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The minimal hardware requirements are as follows,

- | | |
|--------------------|--------------|
| 1. Processor | : Pentium IV |
| 2. RAM | : 8 GB |
| 3. Processor | : 2.4 GHz |
| 4. Main Memory | : 8GB RAM |
| 5. Hard Disk Drive | : 1tb |
| 6. Keyboard | : 104 Keys |

3.3 SOFTWARE REQUIREMENTS

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. The minimal software requirements are as follows,

- | | |
|---------------------|-------------|
| 1. Front end | : python |
| 2. Dataset | : csv |
| 3. IDE | :anaconda |
| 4. Operating System | :Windows 10 |

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

As speech is the form of expressing and to connect with the computer world, we define speech emotion recognition system. It includes grouping together various methodologies to analyze speech signals so that embedded emotions are identified. There are many models available to process speech signals to make prediction and identify the embedded emotion lying in it. Here in this project, we are implementing recurrent neural network model including Long Short Term Memory.

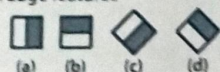
Audio file is a sequential data and needed a appropriate model to learn it and analyze. So we choose sequential lstm model to implement this project. For the facial emotion recognition we use haar cascade algorithm.

4.1. Selected Methodology or process model

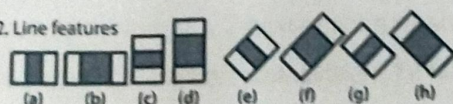
4.1.1 HAAR CASCADE ALGORITHM

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video.

1. Edge features



2. Line features



3. Center-surround features



Face Detection determines the locations and sizes of human faces in arbitrary (digital) images.

In **Face Recognition**, the use of Face Detection comes first to determine and isolate a face before it can be recognized.

Fig 4.1 Haar Cascade Algorithm

We use haar-cascade algorithm and cnn for the emotion detection by facial expression

ARCHITECTURE :

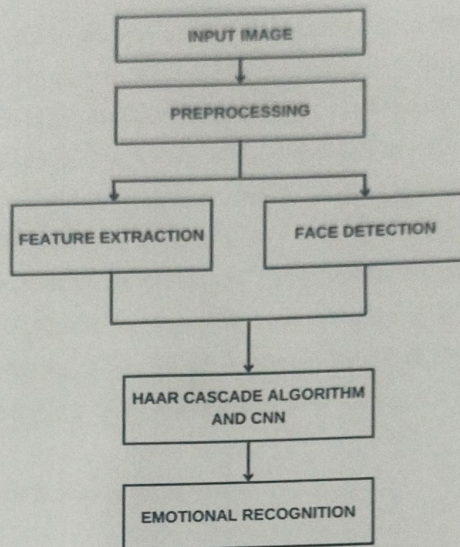


Fig 4.2 Architecture of Haar Cascade Algorithm

4.1.2 MODULE DESCRIPTION

4.1.2.1. Speech Corpus

We use Speech sound records (16bit, 48kHz.wav) from the "RAVDESS" dataset. This bit of the RAVDESS contains 1440 records: 60preliminaries for every entertainer x 24 on-screen characters = 1440. The RAVDESS contains 24 expert on-screen characters "12 female" and "12 male", vocalizing two lexically- coordinated explanations in a nonpartisan "North American" inflection. The emotions were labelled as follows: 01- 'neutral', 02-'calm', 03- 'happy', 04-'sad', 05-'angry', 06- 'fearful', 07-'disgust', 08 -'surprised'.

4.1.2.2.Extraction of features using MFCC

Bringing out the essential attributes is the foremost and the most important step to get the accurate output. It is important to segregate the important and useless attributes from the speech. Important attributes are pitch, tone, etc. Useless features are background noise, disturbances, etc.

Human mouth organs like vocal tract, teeth, gums, tongue play an important role in changing the sounds of our speech. Different forms of sound are created because of different shapes of these organs. To get an accurate data of a speech it is important to study the impact of the varying shapes of these human organs. We can interpret the shape of the vocal tract by marking the frequencies and various other attributes.

Steps involved in working of MFCC:

Energy increment: This is the first step. In this process the energy of the input voice signal is increased by passing it through a filter bank.

Framework: The signals of a sound alter at a very rapid rate when a person is speaking. It is presumed that on a brief time span there is no change in the signals. For this reason, we break the signals into shorter time intervals like 20-40 milliseconds. An extremely shorter time interval is undesirable because then we will require more examples to get a good output. If the time interval is extended that is also undesirable because then the signal will vary a lot in the time frame.

Overlapping: After building up of frameworks, the next task is to overlap the consecutive frameworks. This is done in order to make the signal continuous in nature.

Computation: Computation or calculation of power range is the next task. The idea for this mechanism is taken from the working of an ear part that is, cochlea. As a person speaks this organ starts vibrating at different spots. The frequencies approaching sound waves decide these spots. The periodogram gauge it carries out a similar job that is extremely important. It makes out the amount of vitality present around in various frequencies approaching towards

it. It is helpful in studying the various frequency regions. This step often makes use of Fast Fourier transformations.

Mel scaling: The "mel scale" is used to get an idea about the building of the gauze banks. These gauzes are of different sizes. The initial one is very narrow that gives the idea about the vitality near to 0 Hertz. As the frequencies are escalated the gauzes get broader to give accurate information about energy at higher frequencies.

Applying Logarithmic function: The next step is to take out their logarithm. In order to understand the audio correctly we need to double the recognized volume of audio. This implies huge variations in energy may not be all that varying if the sound is loud enough.

Discrete Cosine Transformation (DCT): The last step is to process the DCT of the log filter bank energies. The range of its coefficients is between 3-14. There are various other coefficients that are not required. They are disposed off.

4.1.2.3. Classifier

Recurrent Neural Network – Long Short Term Memory (LSTM) are used to model tasks where the output is contingent on multiple inputs and dependent on these inputs. Unlike traditional neural networks, RNN operates on a sequence of data inputs and stores the information of the previous inputs. For example, while you are reading a text you don't process individual words, rather you try to remember the previous words and sentences to update the incoming information. In this way you gather a better understanding of the text. RNN works in a similar way. In a speech emotion recognition model, a sequence of voice signals is input in the model and a function is executed at each time step, thus it maintains a state all through.

4.1.3 Python Language

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

Python Programming Characteristics

- It provides rich data types and easier to read syntax than any other programming languages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Python are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

4.1.3.1 Applications of Python Programming

Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use syntax.

About Opencv Package

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several

other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

4.1.4 Features of Anaconda Navigator

Anaconda

Anaconda is a free and open source, easy to install distribution of Python and R programming languages. Anaconda provides a working environment which is used for scientific computing, data science, statistical analysis and machine learning.

The latest distribution of Anaconda is Anaconda 5.3 and is released in October, 2018. It has the conda package, environment manager and a collection of 1000+ open source packages long with free community support.

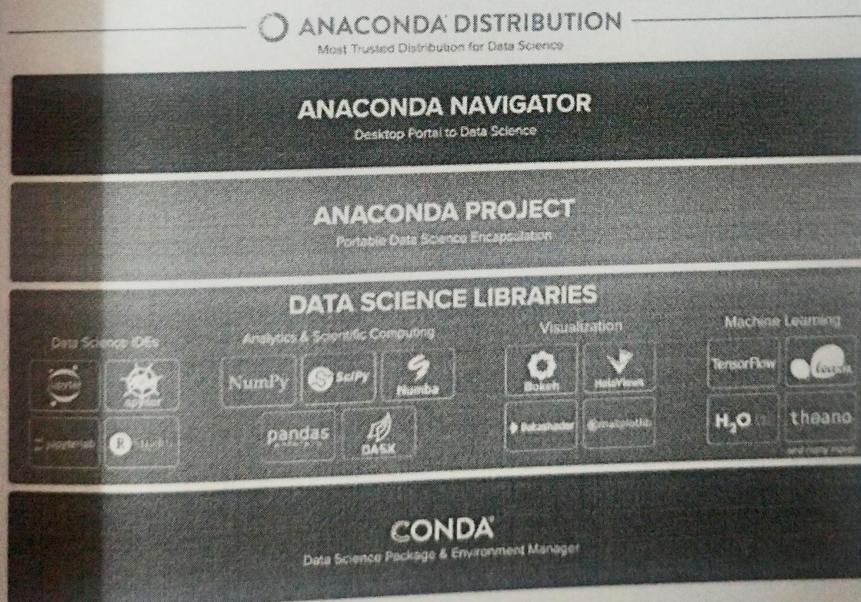
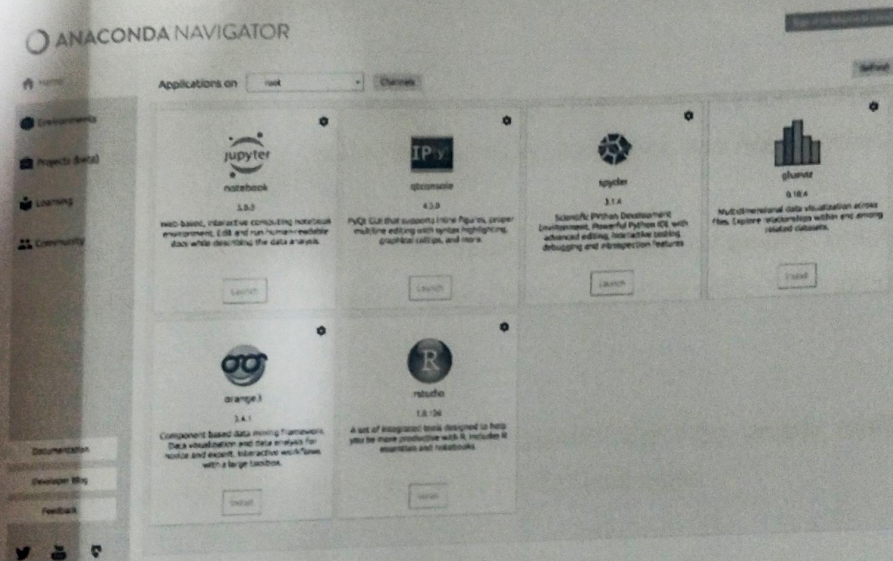


Fig 4.3 Anaconda Navigator

What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.



Anaconda Navigator

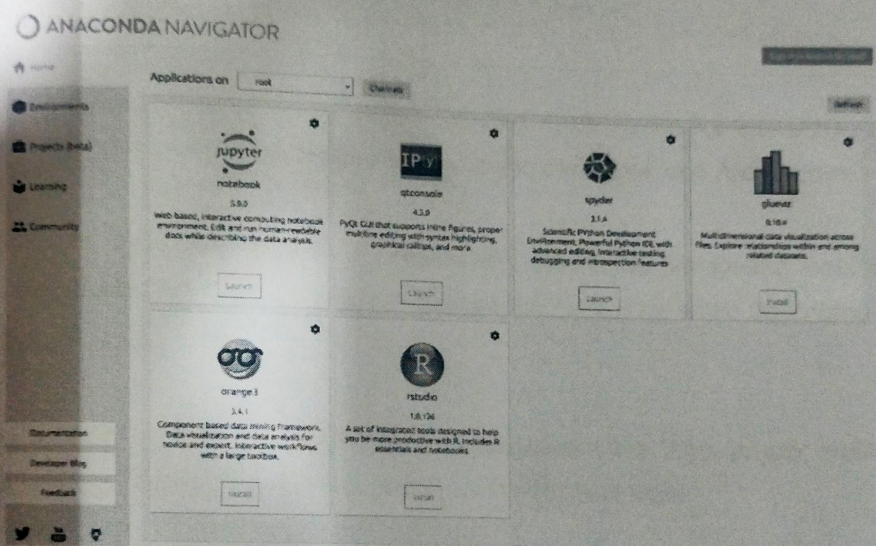
4.1.4.1 Applications Provided In Anaconda Distribution

The Anaconda distribution comes with the following applications along with Anaconda Navigator.

1. JupyterLab
2. Jupyter Notebook
3. Qt Console

What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda distribution. It allows us to launch applications provided in the Anaconda distribution and easily manage conda packages, environments and channels without the use of command-line commands. It is available for Windows, macOS and Linux.



Anaconda Navigator

4.1.4.1 Applications Provided In Anaconda Distribution

The Anaconda distribution comes with the following applications along with Anaconda Navigator.

1. JupyterLab
2. Jupyter Notebook
3. Qt Console

4. Spyder

5. Glueviz

6. Orange3

7. RStudio

8. Visual Studio Code

> JupyterLab: This is an extensible working environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

> Jupyter Notebook: This is a web-based, interactive computing notebook environment. We can edit and run human-readable docs while describing the data analysis.

> Qt Console: It is the PyQt GUI that supports inline Figs, proper multiline editing with syntax highlighting, graphical calltips and more.

> Spyder: Spyder is a scientific Python Development Environment. It is a powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

> VS Code: It is a streamlined code editor with support for development operations like debugging, task running and version control.

> Glueviz: This is used for multidimensional data visualization across files. It explores relationships within and among related datasets.

> Orange 3: It is a component-based data mining framework. This can be used for data visualization and data analysis. The workflows in Orange 3 are very interactive and provide a large toolbox.



4.2 SYSTEM ARCHITECTURE

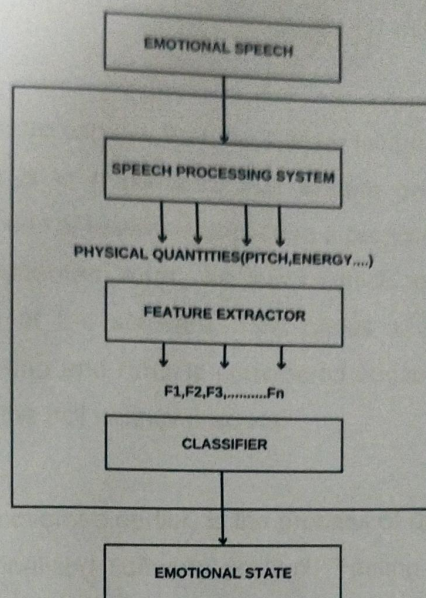


Fig 4.4 System Architecture for speech emotion detection

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 TESTING

General

In a generalized way, we can say that the system testing is a type of testing in which the main aim is to make sure that system performs efficiently and seamlessly. The process of testing is applied to a program with the main aim to discover an unprecedented error, an error which otherwise could have damaged the future of the software. Test cases which brings up a high possibility of discovering and error is considered successful. This successful test helps to answer the still unknown errors.

TEST CASE

Testing, as already explained earlier, is the process of discovering all possible weak-points in the finalized software product. Testing helps to counter the working of sub-assemblies, components, assembly and the complete result. The software is taken through different exercises with the main aim of making sure that software meets the business requirement and user-expectations and doesn't fails abruptly. Several types of tests are used today. Each test type addresses a specific testing requirement.

Testing Techniques

A test plan is a document which describes approach, its scope, its resources and the schedule of aimed testing exercises. It helps to identify almost other test item, the features which are to be tested, its tasks, how will everyone do each task, how much the tester is independent, the environment in which the test is taking place, its technique of design plus the both the end criteria which is used, also rational of choice of theirs, and whatever kind of risk which requires emergency planning. It can be also referred to as the record of the process of test planning. Test plans are usually prepared with signification input from test engineers.

5.1.1 UNIT TESTING

In unit testing, the design of the test cases is involved that helps in the validation of the internal program logic. The validation of all the decision branches and internal code takes place. After the individual unit is completed it takes place. Plus it is taken into account after the individual unit is completed before integration. The unit test thus performs the basic level test at its component stage and test the particular business process, system configurations etc. The unit test ensures that the particular unique path of the process gets performed precisely to the documented specifications and contains clearly defined inputs with the results which are expected.

5.1.2 INTEGRATION TESTING

These tests are designed to test the integrated software items to determine whether if they really execute as a single program or application. The testing is event driven and thus is concerned with the basic outcome of field. The Integration tests demonstrate that the components were individually satisfaction, as already represented by successful unit testing, the components are apt and fine. This type of testing is specially aimed to expose the issues that come-up by the components combination.

5.1.3 FUNCTIONAL TESTING

The functional tests help in providing the systematic representation that functions tested are available and specified by technical requirement, documentation of the system and the user manual.

5.1.4 SYSTEM TESTING

System testing, as the name suggests, is the type of testing in which ensure that the software system meet the business requirements and aim. Testing of the configuration is taken place here to ensure predictable result and thus analysis of it. System testing is relied on the description of process and its flow, stressing on pre driven process and the points of integration.

5.1.5 WHITE BOX TESTING

The white box testing is the type of testing in which the internal components of the system software is open and can be processed by the tester. It is therefore a complex type of testing process. All the data structure, components etc. are tested by the tester himself to find out a possible bug or error. It is used in situation in which the black box is incapable of finding out a bug. It is a complex type of testing which takes more time to get applied.

5.1.6 BLACK BOX TESTING

The black box testing is the type of testing in which the internal components of the software is hidden and only the input and output of the system is the key for the tester to find out a bug. It is therefore a simple type of testing. A programmer with basic knowledge can also process this type of testing. It is less time consuming as compared to the white box testing. It is very successful for software which are less complex are straight-forward in nature. It is also less costly than white box testing.

5.1.7 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

CHAPTER 6

RESULT& DISCUSSIONS

The field of emotion recognition is increasingly relying on deep learning algorithms, which have shown promising results. As this technology advances, it is expected that more dependable systems will emerge, opening up countless applications across various industries. In a recent study, researchers investigated several databases and utilized LSTM to tackle the challenge of emotion recognition. The study employed Tensor Flow to train the model, achieving an accuracy rate of 92.88%. With the same architecture, there is potential to develop a real-time emotion recognition system in the future. Overall, the use of deep learning algorithms in emotion recognition holds great promise for future advancements, and the results of this study demonstrate the potential for accurate and efficient models

CHAPTER 7

CONCLUSION

Deep learning's use of speech and face to identify emotions has shown encouraging results recently and has the potential to revolutionise a number of applications, including customer service, mental health monitoring, and human-robot interaction. Deep learning models are able to learn to extract pertinent features from speech and facial data and classify emotions with high accuracy, enabling the detection of emotions in real-time and under various circumstances. Even so, there are still a number of issues to be resolved, such as the scarcity of datasets, data variability, feature extraction, interpretability, and ethical and privacy issues. A multidisciplinary strategy combining developments in deep learning, signal processing, psychology, ethics, and privacy will be necessary to address these problems.

In conclusion, speech and face emotion detection in deep learning has enormous potential for enhancing human-machine interaction and comprehending human emotions. We can anticipate seeing more advanced and precise models that can recognize a wider range of emotions and be used in a growing number of applications as research and development on this topic continues, while also addressing the ethical and privacy issues raised by this technology.

REFERENCES

- [1] E. M. Albornoz, M. Sanchez-Gutierrez, F. Martinez- Licona , H. L. Rufiner, J. Goddard, Spoken emotion recognition using deep learning, Proceedings of 19th Ibero american Congress on Pattern Recognition, 104-111, 2020.
- [2] H. Zhang, H. Huang and H. Han, "Attention based convolution skip bidirectional long short term memory network for speech emotion recognition", IEEE Access, vol. 9, pp. 5332-5342, 2021.
- [3] H. Chen, D. Jiang and H. Sahli, "Transformer encoder with multi-modal multi-head attention for continuous affect recognition", IEEE Trans. Multimedia, vol. 23.
- [4] N.-H. Ho, H.-J. Yang, S.-H. Kim and G. Lee, "Multimodal approach of speech emotion recognition using multi-level multi-head fusion attention-based recurrent neural network", IEEE Access, vol. 8, pp. 61672-61686, 2020.
- [5] S. Lee, D. K. Han and H. Ko, "Multimodal emotion recognition fusion analysis adapting BERT with heterogeneous feature unification", IEEE Access, vol. 9, pp. 94557-94572, 2021.
- [6] T. Baltrušaitis, C. Ahuja and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy", IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 2, pp. 423-443, Feb. 2019.
- [7] B T Atmaja and M Akagi, "Speech emotion recognition based on speech segment using lstm with attention model[C]", 2019 IEEE International Conference on Signals and Systems (ICSigSys)., pp. 40-44, 2019.
- [8] S Mao, D Tao, G Zhang, P C Ching and T Lee, "Revisiting hidden Markov models for speech emotion recognition[C]", ICASSP 2019-2019 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)., pp. 6715-6719, 2019.
- [9] Leonardo Pepino, Pablo Riera, Luciana Ferrer and Agustín Gravano, "Fusion approaches for emotion recognition from speech using acoustic and text-based features", ICASSP 2020 - 2020 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 6484-6488, 2020.

- [10] S. Lin, "Robust pitch estimation and tracking for speakers based on subband encoding and the generalized labeled multi-bernoulli filter", *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 4, pp. 827-841, Apr. 2019.
- [11] Kwon, S., 2019. A CNN-assisted enhanced audio signal processing for speech emotion recognition. *Sensors*, 20(1), p.183.
- [12] Busso, C., Bulut, M., Lee, C.C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J.N., Lee, S. and Narayanan, S.S., 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4), pp.335-359.
- [13] Tariq, Z., Shah, S.K. and Lee, Y., 2019, December. Speech emotion detection using iot based deep learning for health care. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 4191-4196). IEEE.
- [14] Livingstone, S.R. and Russo, F.A., 2018. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PloS one*, 13(5), p.e0196391.
- [15] Rabiner, L.R. and Schafer, R.W., 2007. *Introduction to digital speech processing* (Vol. 1). Now Publishers Inc.

APPENDIX

SOURCE CODE

Model creation

```
import time
import os
import numpy as np

## Audio Preprocessing ##
import pyaudio
import wave
import librosa
from scipy.stats import zscore

## Time Distributed CNN ##
import tensorflow as tf
from keras import backend as K
from keras.models import Model
from keras.layers import Input, Dense, Dropout, Activation, TimeDistributed
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Flatten
from keras.layers import LSTM

'''
Speech Emotion Recognition
'''
class speechEmotionRecognition:

    '''
    Voice recording function
    '''
    def __init__(self, subdir_model=None):

        # Load prediction model
        if subdir_model is not None:
            self._model = self.build_model()
            self._model.load_weights(subdir_model)

        # Emotion encoding
        self._emotion = {0:'Angry', 1:'Disgust', 2:'Fear', 3:'Happy', 4:'Neutral', 5:'Sad',
        6:'Surprise'}

    '''
    Voice recording function
    '''
    def voice_recording(self, filename, duration=5, sample_rate=16000,
    chunk=1024, channels=1):
```



```

# Start the audio recording stream
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paInt16,
                 channels=channels,
                 rate=sample_rate,
                 input=True,
                 frames_per_buffer=chunk)

# Create an empty list to store audio recording
frames = []

# Determine the timestamp of the start of the response interval
print('* Start Recording *')
stream.start_stream()
start_time = time.time()
current_time = time.time()

# Record audio until timeout
while (current_time - start_time) < duration:

    # Record data audio data
    data = stream.read(chunk)

    # Add the data to a buffer (a list of chunks)
    frames.append(data)

    # Get new timestamp
    current_time = time.time()

# Close the audio recording stream
stream.stop_stream()
stream.close()
p.terminate()
print('* End Recording *')

# Export audio recording to wav format
wf = wave.open(filename, 'w')
wf.setnchannels(channels)
wf.setsampwidth(p.get_sample_size(paudio.paInt16))
wf.setframerate(sample_rate)
wf.writeframes(b''.join(frames))
wf.close()

'''
Mel-spectrogram computation
'''
def mel_spectrogram(self, y, sr=16000, n_fft=512, win_length=256,
                    hop_length=128, window='hamming', n_mels=128, fmax=4000):

```



```

    # Compute spectrogram
    mel_spect = np.abs(librosa.stft(y, n_fft=n_fft, window=window,
    win_length=win_length, hop_length=hop_length)) ** 2

    # Compute mel spectrogram
    mel_spect = librosa.feature.melspectrogram(S=mel_spect, sr=sr,
    n_mels=n_mels, fmax=fmax)

    # Compute log-mel spectrogram
    mel_spect = librosa.power_to_db(mel_spect, ref=np.max)

    return np.asarray(mel_spect)

'''
Audio framing
'''
def frame(self, y, win_step=64, win_size=128):

    # Number of frames
    nb_frames = 1 + int((y.shape[2] - win_size) / win_step)

    # Framming
    frames = np.zeros((y.shape[0], nb_frames, y.shape[1],
    win_size)).astype(np.float16)
    for t in range(nb_frames):
        frames[:,t,:,:] = np.copy(y[:,:(t * win_step):(t * win_step +
    win_size)]).astype(np.float16)

    return frames

'''
Time distributed Convolutional Neural Network model
'''
def build_model(self):

    # Clear Keras session
    K.clear_session()

    # Define input
    input_y = Input(shape=(5, 128, 128, 1), name='Input_MELSPECT')

    # First LFLB (local feature learning block)
    y = TimeDistributed(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),
    padding='same'), name='Conv_1_MELSPECT')(input_y)
    y = TimeDistributed(BatchNormalization(),
    name='BatchNorm_1_MELSPECT')(y)
    y = TimeDistributed(Activation('elu'), name='Activ_1_MELSPECT')(y)

```



```

y = TimeDistributed(MaxPooling2D(pool_size=(2, 2), strides=(2, 2),
padding='same'), name='MaxPool_1_MELSPECT')(y)
y = TimeDistributed(Dropout(0.2), name='Drop_1_MELSPECT')(y)

# Second LFLB (local feature learning block)
y = TimeDistributed(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),
padding='same'), name='Conv_2_MELSPECT')(y)
y = TimeDistributed(BatchNormalization(),
name='BatchNorm_2_MELSPECT')(y)
y = TimeDistributed(Activation('elu'), name='Activ_2_MELSPECT')(y)
y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4),
padding='same'), name='MaxPool_2_MELSPECT')(y)
y = TimeDistributed(Dropout(0.2), name='Drop_2_MELSPECT')(y)

# Third LFLB (local feature learning block)
y = TimeDistributed(Conv2D(128, kernel_size=(3, 3), strides=(1, 1),
padding='same'), name='Conv_3_MELSPECT')(y)
y = TimeDistributed(BatchNormalization(),
name='BatchNorm_3_MELSPECT')(y)
y = TimeDistributed(Activation('elu'), name='Activ_3_MELSPECT')(y)
y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4),
padding='same'), name='MaxPool_3_MELSPECT')(y)
y = TimeDistributed(Dropout(0.2), name='Drop_3_MELSPECT')(y)

# Fourth LFLB (local feature learning block)
y = TimeDistributed(Conv2D(128, kernel_size=(3, 3), strides=(1, 1),
padding='same'), name='Conv_4_MELSPECT')(y)
y = TimeDistributed(BatchNormalization(),
name='BatchNorm_4_MELSPECT')(y)
y = TimeDistributed(Activation('elu'), name='Activ_4_MELSPECT')(y)
y = TimeDistributed(MaxPooling2D(pool_size=(4, 4), strides=(4, 4),
padding='same'), name='MaxPool_4_MELSPECT')(y)
y = TimeDistributed(Dropout(0.2), name='Drop_4_MELSPECT')(y)

# Flat
y = TimeDistributed(Flatten(), name='Flat_MELSPECT')(y)

# LSTM layer
y = LSTM(256, return_sequences=False, dropout=0.2,
name='LSTM_1')(y)

# Fully connected
y = Dense(7, activation='softmax', name='FC')(y)

# Build final model
model = Model(inputs=input_y, outputs=y)

return model

```



```

'''
Predict speech emotion over time from an audio file
'''
def predict_emotion_from_file(self, filename, chunk_step=16000,
chunk_size=49100, predict_proba=False, sample_rate=16000):

    # Read audio file
    y, sr = librosa.core.load(filename, sr=sample_rate, offset=0.5)

    # Split audio signals into chunks
    chunks = self.frame(y.reshape(1, 1, -1), chunk_step, chunk_size)

    # Reshape chunks
    chunks = chunks.reshape(chunks.shape[1], chunks.shape[-1])

    # Z-normalization
    y = np.asarray(list(map(zscore, chunks)))

    # Compute mel spectrogram
    mel_spect = np.asarray(list(map(self.mel_spectrogram, y)))

    # Time distributed Framing
    mel_spect_ts = self.frame(mel_spect)

    # Build X for time distributed CNN
    X = mel_spect_ts.reshape(mel_spect_ts.shape[0],
mel_spect_ts.shape[1],
mel_spect_ts.shape[2],
mel_spect_ts.shape[3],
1)

    # Predict emotion
    if predict_proba is True:
        predict = self._model.predict(X)
    else:
        predict = np.argmax(self._model.predict(X), axis=1)
        predict = [self._emotion.get(emotion) for emotion in predict]

    # Clear Keras session
    K.clear_session()

    # Predict timestamp
    timestamp = np.concatenate([[chunk_size], np.ones((len(predict) - 1)) *
chunk_step]).cumsum()
    timestamp = np.round(timestamp / sample_rate)

    return [predict, timestamp]

'''
Export emotions predicted to csv format
'''

```



```

'''
def prediction_to_csv(self, predictions, filename, mode='w'):

    # Write emotion in filename
    with open(filename, mode) as f:
        if mode == 'w':
            f.write("EMOTIONS"+"\n")
            for emotion in predictions:
                f.write(str(emotion)+"\n")
    f.close()

```

Web application

```

#!/usr/bin/python3
# -*- coding: utf-8 -*-

```

```

#### General imports ####
from __future__ import division
import numpy as np
import pandas as pd
import time
import re
import os
from collections import Counter

```

```

#### Flask imports
import requests
from flask import Flask, render_template, session, request, redirect, flash,
Response

```

```

#### Audio imports ####
from speechEmotionRecognition import *

```

```

# Flask config
app = Flask(__name__)
app.secret_key = b'(\xee\x00\xd4\xce"\xcfx\xe8@\r\xde\xfc\xbdJ\x08W'
app.config['UPLOAD_FOLDER'] = '/Upload'
path=r"D:\Gowtham\Project\Speech_Emotion\Speech_Emotion\Speech_Emot
ion_Detection-using-LSTM-model-main\Speech Emotion Detection In
Flask\Upload"
#####
#####
##### INDEX
#####
#####
#####

```

```

@app.route('/', methods=['GET'])
def index():

```



```

return render_template('index.html')

#####
#####
##### AUDIO Prediction
#####
#####

# Audio Index
@app.route('/audio_index', methods=['POST'])
def audio_index():
    return render_template('audio.html', display_button=False)

# Audio Recording
@app.route('/audio_recording', methods=("POST", "GET"))
def audio_recording():

    # Instantiate new SpeechEmotionRecognition object
    SER = speechEmotionRecognition()

    # Voice Recording
    rec_duration = 10 # in sec
    rec_sub_dir = os.path.join(path, 'voice_recording.wav')
    SER.voice_recording(rec_sub_dir, duration=rec_duration)

    # Send Flash message
    flash("The recording is over! You now have the opportunity to do an
analysis of your emotions. If you wish, you can also choose to record yourself
again.")

    return render_template('audio.html', display_button=True)

# Audio Emotion Analysis
@app.route('/audio_analysis', methods=("POST", "GET"))
def audio_analysis():

    # Sub dir to speech emotion recognition model
    model_sub_dir =
os.path.join(r'C:\Users\DELL\Music\Speech_Emotion_Detection-using-LSTM-
model-main\Speech_Emotion_Detection-using-LSTM-model-main\Speech
Emotion Detection In Flask\Models', 'audio.hdf5')

    # Instantiate new SpeechEmotionRecognition object
    SER = speechEmotionRecognition(model_sub_dir)

    # Voice Record sub dir
    rec_sub_dir = os.path.join(path, 'voice_recording.wav')

```



```

# Predict emotion in voice at each time step
step = 1 # in sec
sample_rate = 16000 # in kHz
emotions, timestamp = SER.predict_emotion_from_file(rec_sub_dir,
chunk_step=step*sample_rate)

```

```

# Export predicted emotions to .txt format
SER.prediction_to_csv(emotions,
os.path.join(r"C:\Users\DELL\Music\Speech_Emotion_Detection-using-LSTM-
model-main\Speech_Emotion_Detection-using-LSTM-model-main\Speech
Emotion Detection In Flask\static", "js", "audio_emotions.txt"), mode='w')
SER.prediction_to_csv(emotions,
os.path.join(r"C:\Users\DELL\Music\Speech_Emotion_Detection-using-LSTM-
model-main\Speech_Emotion_Detection-using-LSTM-model-main\Speech
Emotion Detection In Flask\static", "js", "audio_emotions_other.txt"), mode='a')

```

```

# Get most common emotion during the interview
major_emotion = max(set(emotions), key=emotions.count)

```

```

# Calculate emotion distribution
emotion_dist = [int(100 * emotions.count(emotion) / len(emotions)) for emotion
in SER._emotion.values()]

```

```

# Export emotion distribution to .csv format for D3JS
df = pd.DataFrame(emotion_dist, index=SER._emotion.values(),
columns=["VALUE"]).rename_axis('EMOTION')

```

```

df.to_csv(os.path.join(r'D:\Gowtham\Project\Speech_Emotion\Speech_Emotio
n\Speech_Emotion_Detection-using-LSTM-model-main\Speech Emotion
Detection In Flask\static', 'js', 'audio_emotions_dist.txt'), sep=',')

```

```

df_other =
pd.read_csv(os.path.join(r"D:\Gowtham\Project\Speech_Emotion\Speech_Em
otion\Speech_Emotion_Detection-using-LSTM-model-main\Speech Emotion
Detection In Flask\static", "js", "audio_emotions_other.txt"), sep=",")

```

```

major_emotion_other = df_other.EMOTION.mode()[0]

```

```

emotion_dist_other = [int(100 * len(df_other[df_other.EMOTION==emotion]) /
len(df_other)) for emotion in SER._emotion.values()]

```

```

# Export emotion distribution to .csv format for D3JS
df_other = pd.DataFrame(emotion_dist_other, index=SER._emotion.values(),
columns=["VALUE"]).rename_axis('EMOTION')

```

```

df_other.to_csv(os.path.join(r'D:\Gowtham\Project\Speech_Emotion\Speech_
Emotion\Speech_Emotion_Detection-using-LSTM-model-main\Speech
Emotion Detection In Flask\static', 'js', 'audio_emotions_dist_other.txt'), sep=',')

```



```
#emotion_dist_other[4]=emotion_dist_other[4]-30  
return render_template('audio_analysis.html', emo=major_emotion,  
emo_other=major_emotion_other, prob=emotion_dist,  
prob_other=emotion_dist_other)
```

```
if __name__ == '__main__':  
    app.run(debug=True)
```


SCREENSHOTS:

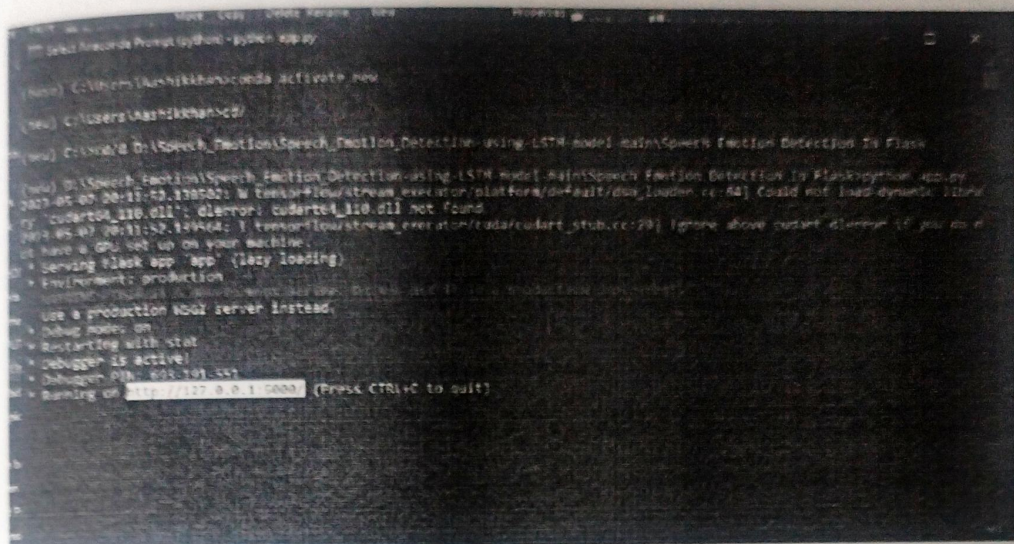


Figure 1 Opening the software using anaconda prompt

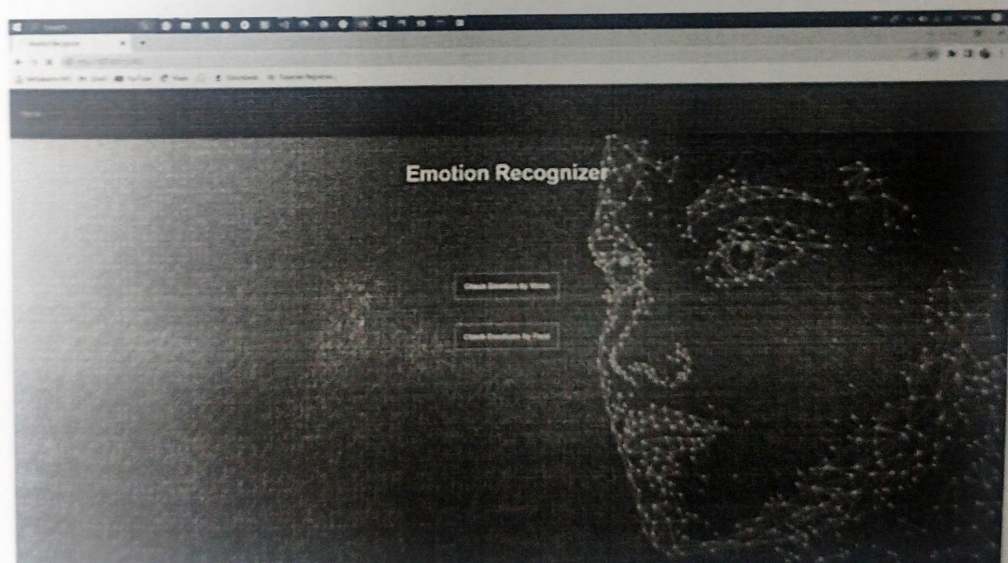


Figure 2 After copying the link provided from the anaconda prompt paste it in a chrome page

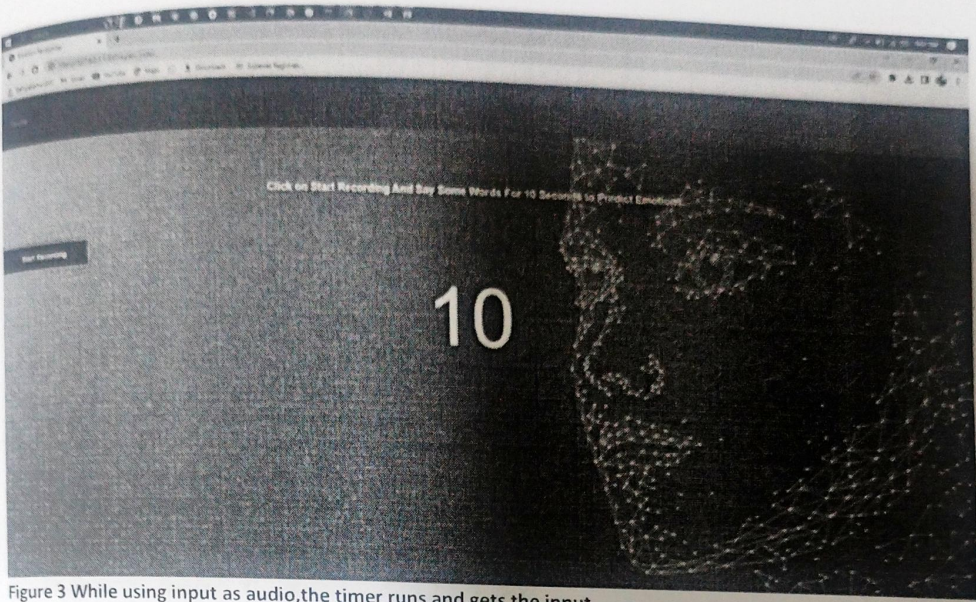


Figure 3 While using input as audio,the timer runs and gets the input

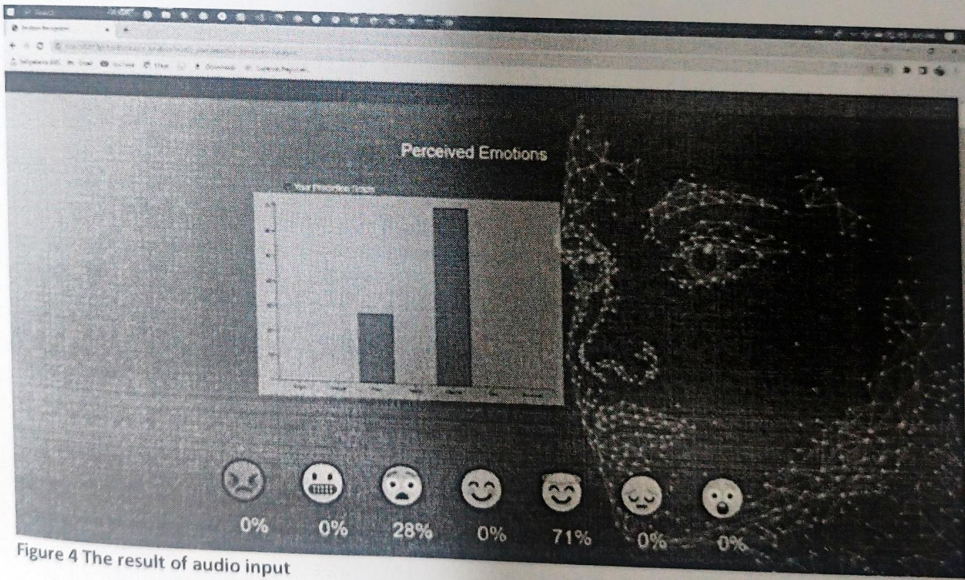


Figure 4 The result of audio input

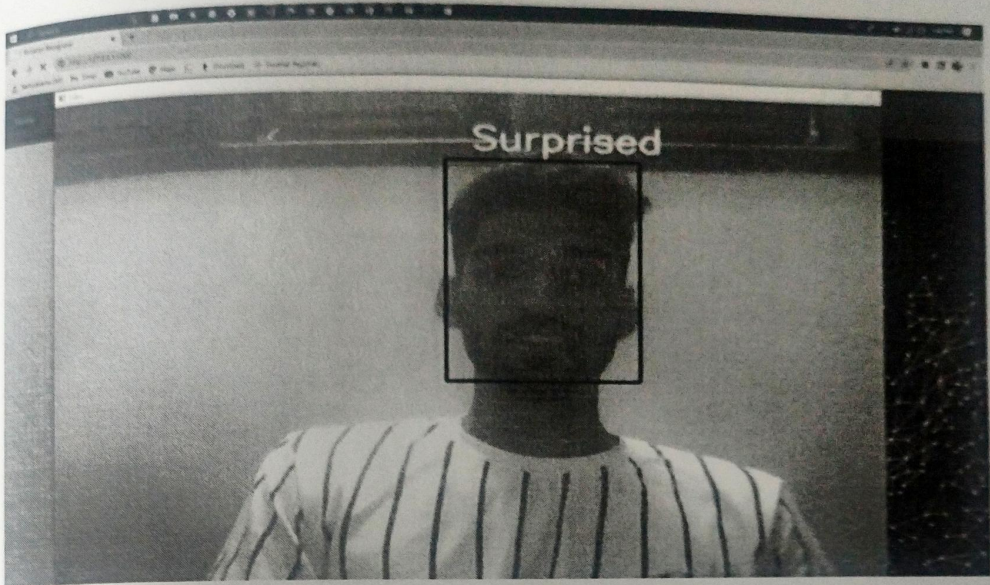


Figure 5 The outcome of a video input