

**A
Project Report
On
“Application of Emotional Model and Use of Mood
and Tone for Speech Synthesis”**

Prepared by
Dhvanit Aghara (16IT001)
Fenil Bhanavadiya (16IT006)
Divya Patel (16IT069)

Under the guidance of

Prof. Hemant Yadav

A Report Submitted to
Charotar University of Science and Technology
for Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology
in Information Technology
(8th Semester Software Project Major-IT407)

Submitted at



**SMT. KUNDANBEN DINSHA PATEL DEPARTMENT OF
INFORMATION TECHNOLOGY**

Chandubhai S. Patel Institute of Technology

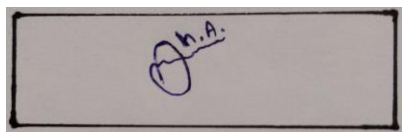
At: Changa, Dist: Anand – 388421

April 2020

CANDIDATE'S DECLARATION

We hereby declare that the project entitled “**Application of Emotional Model and Use of Mood and Tone for Speech Synthesis**” is our own work conducted under the guidance of **Prof. Hemant Yadav** and **Dr. Abdul Jhummarwala**.

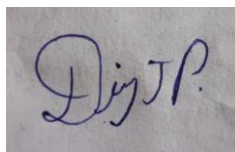
We further declare that to the best of our knowledge, the project for B. Tech does not contain any part of the work, which has been submitted for the award of any degree either in this University or in other University without proper citation.

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'Dh.A.' with a stylized flourish.

Dhvanit Aghara (16IT001)

A rectangular box containing a handwritten signature in blue ink. The signature is 'Fenil' with a horizontal line underneath.

Fenil Bhanavadiya (16IT006)

A rectangular box containing a handwritten signature in blue ink. The signature is 'Divya P.' with a stylized flourish.

Divya Patel (16IT069)

Prof. Hemant Yadav
Assistant Professor,
Smt. Kundanben Dinsha Patel Department of Information Technology,
Faculty of Technology & Engineering,
Changa – 388425.

CERTIFICATE

*This is to certify that the project report compiled by **Mr. Dhvanit H. Aghara, Mr. Fenil N. Bhanavadiya and Mr. Divya J. Patel** students of 8th Semester **B-Tech-IT** from **Chandubhai S. Patel Institute of Technology, CHARUSAT, Changa** have completed their final semester project satisfactorily. To the best of our knowledge this is an original and bonafide work done by them. They have worked on machine learning-based application “**Application of Emotional Model and Use of Mood and Tone for Speech Synthesis**”, starting from December 16th, 2019 to April 30th, 2020.*

During their tenure at this Institute, they were found to be sincere and meticulous in their work. We appreciate their enthusiasm & dedication towards the work assigned to them.

We wish them every success.



Dr. Abdul Jhummarwala
Faculty,

BISAG, Gandhinagar

T. P. Singh
Director,

BISAG, Gandhinagar

CERTIFICATE

This is to certify that the report entitled “**Application of Emotional Model and Use of Mood and Tone for Speech Synthesis**” is a bonafide work carried out by **Mr. Dhvanit H. Aghara (16IT001)**, **Mr. Fenil N. Bhanavadiya (16IT006)** and **Mr. Divya J. Patel (16IT069)** under the guidance and supervision of **Prof. Hemant Yadav & Dr. Abdul Jhummarwala** for the subject **Software Project Major (IT407)** of 8th Semester of Bachelor of Technology in **Information Technology** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,



Prof. Hemant Yadav
Assistant Professor
Smt. Kundanben Dinsha Patel Department
of Information Technology
CSPIT, Changa, Gujarat.

Dr. Abdul Jhummarwala
Faculty
BISAG, Gandhinagar.

Dr. Parth Shah
Head & Associate Professor
Smt. Kundanben Dinsha Patel Department of Information
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

ABSTRACT

Disability of visual text reading has a huge impact on the quality of life for visually disabled people. Although there have been several devices designed for helping visually disabled to see objects using an alternating sense such as sound and touch, but the development of text reading device is still at an early stage.

Therefore, this project “Application of Emotional Model and Use of Mood and Tone for Speech Synthesis” is used to recognise text from image and convert it into speech with emotion which is detected from text. Model is build using python language. It can work on almost all types of image formats including JPEG, PNG, BMP. The solution developed for this project consists of three major parts: Text recognition, Sentiment analyses, Text to speech conversion. User can provide an image having text into it, model will first recognize text and then detect emotion from it (here we used two emotions: happiness and sadness) and then convert text into speech with detected emotion.

ACKNOWLEDGEMENT

Working on this project “Application of Emotional Model and Use of Mood and Tone for Speech Synthesis” was a source of immense knowledge for us. We have found this rare opportunity to evince a word of thanks to all those who played a key role in the successful completion of our project.

We sincerely thank our Head of Department Dr. Parth Shah Sir for giving the chance as well as support for all the time being. And his able guidance and continuous encouragement made us work in all the challenges during project development.

We express deep gratitude to assistant **Prof. Hemant Yadav** internal project guide from Faculty of Engineering, CHARUSAT and **Dr. Abdul Jhummarwala** External project guide from BISAG, Gandhinagar for their valuable suggestions, help and moral support. We also thank to all those who could not find a separate name but have helped directly and indirectly.

Table of Contents

Abstract	ii
Acknowledgement	iii
Chapter 1 : Introduction	1
1.1 Project Overview	2
1.2 Objective	2
1.3 Scope.....	3
1.4 Tool and Technology Used	3
Chapter 2 : Project Management	4
2.1 Project Planning	5
2.1.1 Project Development Approach and Justification.....	5
2.1.2 Project Efforts and Time, Cost Estimation.....	7
2.2 Project Work Scheduling (Gantt Chart/PERT/Network Chart)	8
Chapter 3 : System Requirement Study	10
3.1 User Characteristics.....	11
3.1.1 Use Case Diagram	11
3.2 Hardware and Software Requirement	12
3.3 Assumptions and Constrains	12
Chapter 4 : System Analysis	13
4.1 Study of Existing Solution.....	14
4.2 Limitation of Existing Solution	14
4.3 Requirement of Proposed System	15
4.3.1. Functional Requirements.....	15
4.3.2 Non-Functional Requirements	15
4.4 System Work Flow	16
4.5 Sequence Diagram	17
4.6 State Chart Diagram	18
4.7 Activity Diagram	19
Chapter 5 : System Design	20
5.1 Data Dictionary	21

5.2 Screen Layout (Forms and Reports)	22
5.3 Method Pseudo Code	29
Chapter 6 : System Implementation and Testing.....	32
6.1 Coding Standards	33
6.2 Testing Methods	33
6.3 Test Suits Designs	34
6.4 Test Cases	35
Chapter 7 : Future Enhancement	40
7.1 Future Enhancement	41
Chapter 8 : Conclusion	42
8.1 Self Analysis of Project Viabilities	43
8.2 Problem Encountered and their Solutions	43
8.3 Summary of Project Work	44
References.....	45

List of Figures

Fig 2.1 – Agile Model	5
Fig 2.2 – Gantt Chart	9
Fig 3.1 – Use case Diagram	11
Fig 4.1 – System Work Flow Diagram	16
Fig 4.2 – Sequence Diagram	17
Fig 4.3 – State Chart Diagram	18
Fig 4.4 – Activity Diagram	19
Fig 5.1 – OCR output 1	22
Fig 5.2 – OCR output 2	23
Fig 5.3 – OCR output 3	24
Fig 5.4 – OCR output 4	24
Fig 5.5 – OCR output 5	25
Fig 5.6 – Emotion Classification output 1	26
Fig 5.7 – Emotion Classification output 2	26
Fig 5.8 – Emotion Classification output 3	27
Fig 5.9 – Emotion Classification output 4	27
Fig 5.10 – text to Speech output	28

List of Tables

Table 2.1 – Value of constant for effort calculation	5
Table 2.2 – Value of constant for development time calculation	5
Table 5.1 – Dataset Summary	21
Table 6.1 – OCR Test Case 1	35
Table 6.2 – OCR Test Case 2	36
Table 6.3 – Emotion Classification Test Case	37
Table 6.4 – Generate Output Speech Test Case	38

CHAPTER 1: INTRODUCTION

1.1 PROJECT OVERVIEW

Disability of visual text reading has a huge impact on the quality of life for visually disabled people. Although there have been several devices designed for helping visually disabled to see objects using an alternating sense such as sound and touch, but the development of text reading device is still at an early stage.

Therefore, this project “Application of Emotional Model and Use of Mood and Tone for Speech Synthesis” is used to recognise text from image and convert it into speech with emotion which is detected from text.

The solution developed for this project consists of three major parts: Text recognition, Sentiment analyses, Text to speech conversion.

In this project, we developed a model in which user can provide an image having text into it, model will first recognize text and then detect emotion from it (here we used two emotions: happiness and sadness) and then convert text into speech with detected emotion.

1.2 OBJECTIVE

The main objective of this project is to build an application which can be used to vocalize text from image with emotions. There are some application which can recognise text and enounce it, but this application can also recognise emotion from text and express it with emotion.

Compare different tools and algorithms to get higher accuracy in result. Provide application which can speak out text from image and sounds more realistic.

1.3 SCOPE

Using this application, we can generate emotional speech like human with recognising text from image. Main focus of this project is to help visually disable people to read text from image. This application can be used in many different areas also like teaching, newspaper reading, storytelling, etc.

1.4 TOOLS & TECHNOLOGY

Programming language : Python

IDE : Spyder

OpenCV (library for image processing)

Tesseract (OCR-tool)

NLTK (Natural Language toolkit for emotion classification)

Tacotron (Text-to-Speech synthesis)

Wavenet (Generate new audio based on the emotion)

CHAPTER 2: PROJECT MANAGEMENT

2.1 PROJECT PLANNING

2.1.1 Project Development Approach

"**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

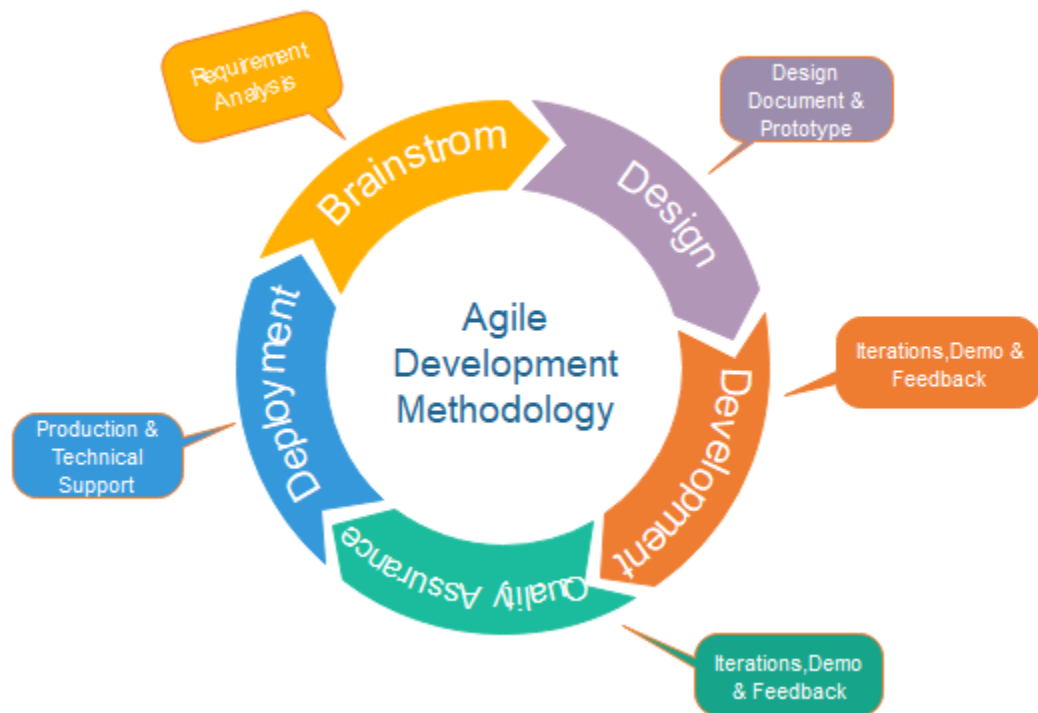


Fig 2.1 Agile Model

Phases of Agile Model:

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Advantages:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.

2.1.2 Project Efforts and Time, Cost Estimation

We have used basic COCOMO estimation technique for the estimation of our project. We concluded to include the project under semidetached category.

The equations for basic COCOMO model are:

$$\text{Effort: } E = aKLOC^b \quad (2.1.2.1)$$

$$\text{Development time: } D = c * (\text{Effort})^d \quad (2.1.2.2)$$

Table 2.1 Value of constant for effort calculation

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Table 2.2 Value of constant for development time calculation

Parameter	Organic	Semi-detached	Embedded
<i>C</i>	2.5	2.5	2.5
<i>D</i>	0.38	0.35	0.32

Effort = $3.0 * (\text{KLOC})^{1.12}$ Person Month.

Development Time = $2.5 * (\text{Effort})^{0.35}$ Months.

Σ Line of Code=2000 (Approx.)

Effort= $3.0 * (2)^{1.12}$ =6.52 Person month.

Development Time= $2.5 * (6.52)^{0.35}$ =4.8 Months(Approx.)

Cost Required to Develop the Product= $4.8 * 2066.16$ = Rs. 9917.56(Approx.)

2.2 PROJECT WORK SCHEDULING

The Project Scheduling's overview is provided by Gantt chart. The whole project planning with all the planning details is shown in the Gantt chart. The Gantt chart will provide the weekly task of the project development.

Gantt Chart

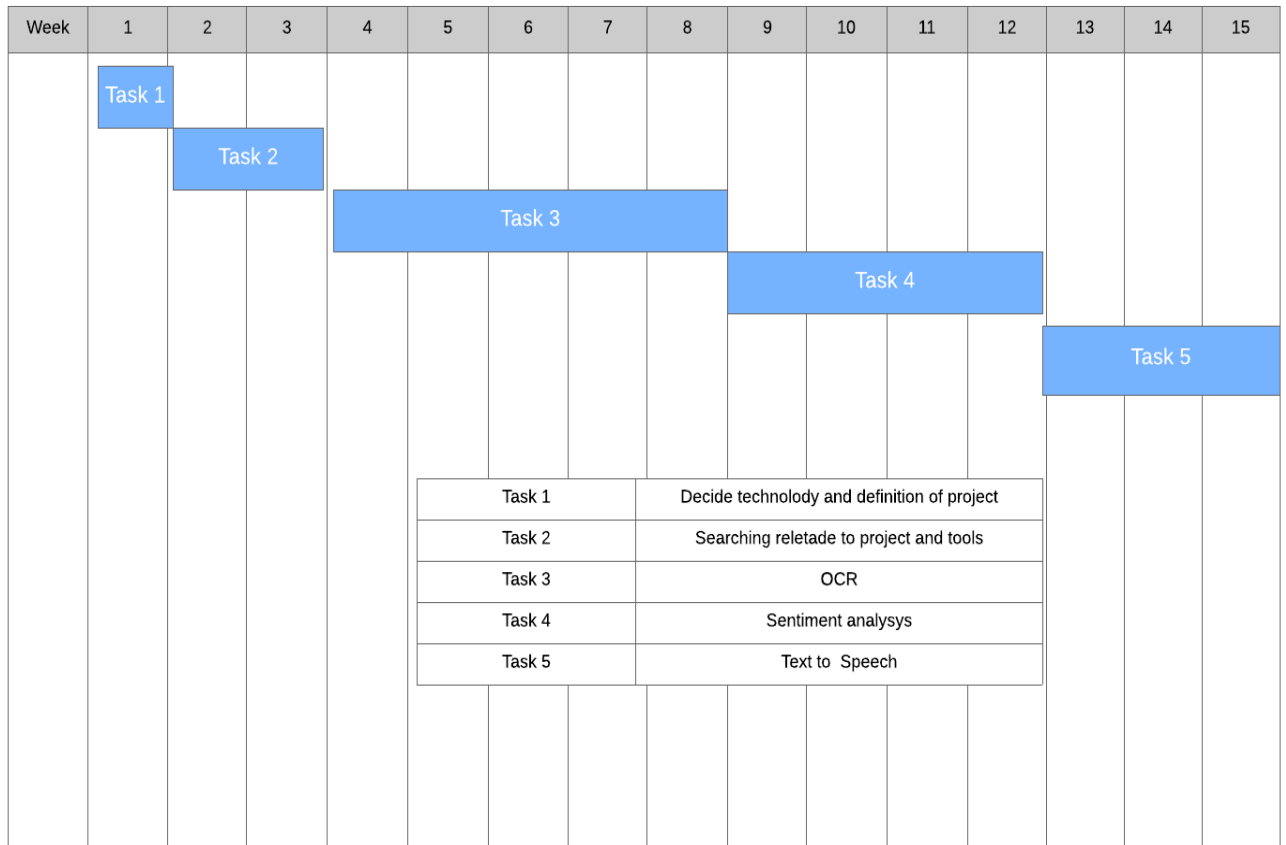


Fig 2.2 Gantt Chart

CHAPTER 3: SYSTEM REQUIREMENTS STUDY

3.1 USER CHARACTERISTIC

Use Case Diagram:

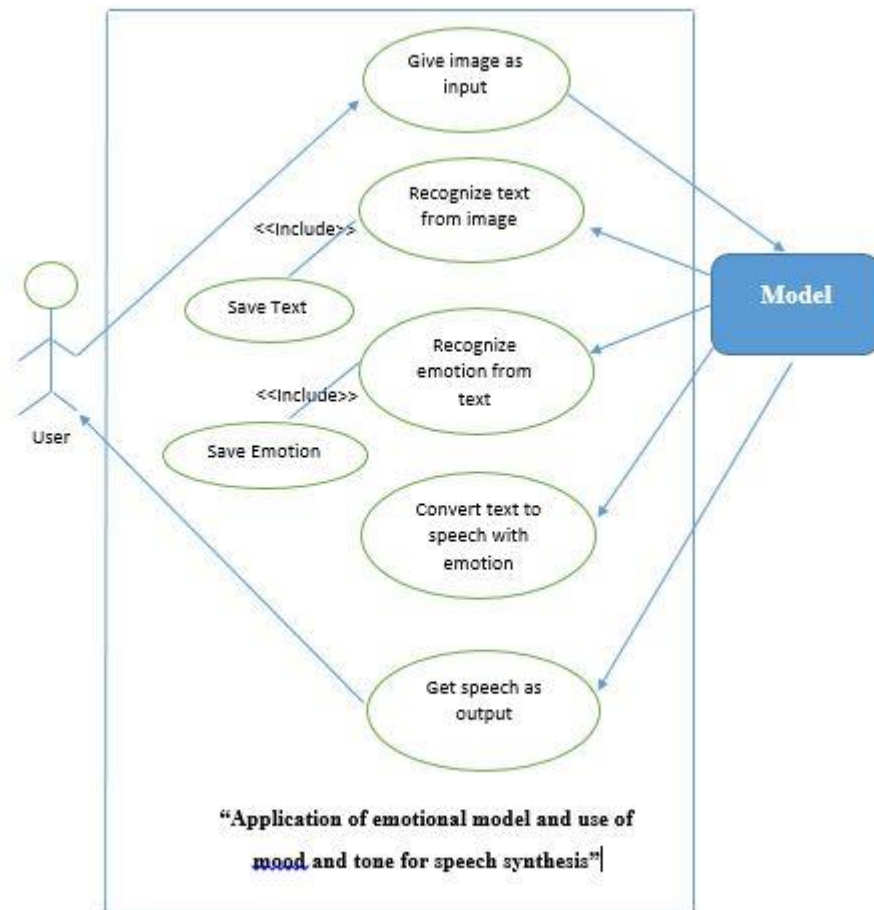


Fig 3.1 Use Case Diagram

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirements:

- RAM: 4 GB
- Hard Disk: 250 GB

Software requirements:

- Python 3.8
- Python IDE
- Python libraries and tools used in project

3.3 ASSUMPTIONS AND CONSTRAINTS

- Minimum Hardware requirements are satisfied.
- Python and necessary tools & libraries are installed.
- Required dataset is available.
- Accessibility: 24 * 7 easily accessible. (click & run)
- Input image must have text in English.
- Image is clear enough to recognize text.
- Reusable code.
- Scalability: We can add more resources to our project without disturbing current scenario.
- Maintenance and testing is done.

CHAPTER 4: SYSTEM ANALYSIS

4.1 STUDY OF EXISTING SOLUTION

- **Existing Solution For OCR**

OpenCV and Tesseract tool.

OpenCV is used for image pre-processing. Tesseract is used for text recognition from processed image. This program works good with large amount of text as well as with some background noise.

OpenCV and Tesseract tool.

OpenCV is used for image pre-processing as well as character detection and segmentation. Tesseract is used for text recognition. In this program we draw bounding boxes around text and give that part to tesseract. This works good with image having more background noise.

OpenCV and TensorFlow.

OpenCV is used for image pre-processing. TensorFlow is used for creating CNN model which recognize text from image. This program works only on word level and only on capital words as model trained on dataset having only capital characters.

4.2 LIMITATIONS OF EXISTING SOLUTION

- **Font Size.** OCR may not convert characters with very large or very small font sizes. This can make the most important characters and words unavailable for text-based systems.
- **Uni-Dimensional.** With OCR, individual words have one dimension, they're either before or after other words. OCR does not catalog page coordinate information for characters even though page coordinates can be quite useful for classification and extracting attributes.

- Sequential Editing. OCR errors typically have to be corrected sequentially with the same errors being repeatedly being edited. Global spell checking can introduce other errors.
- Case Sensitivity for Editing. The use of spell checking to correct OCR text will typically not permit the case of the letters to be considered, e.g., cat and CAT will be treated alike.
- Languages. Many languages have special characters, and unless the correct OCR software is loaded, those characters can be lost or incorrectly recognized.

4.3 REQUIREMENTS OF PROPOSED SYSTEM

4.3.1 Functional Requirements :

- Take text image as input.
- Recognize text from image.
- Classify mood from text.
- Convert text into speech with given emotion.

4.3.2 Non - Functional Requirements :

- Input image must have text in English.
- Accessibility: It can be easily accessible i.e click & run.
- Scalability: we can add more resources to our project without disturbing current scenario.
- Improve accuracy of model.

4.4 SYSTEM WORK FLOW

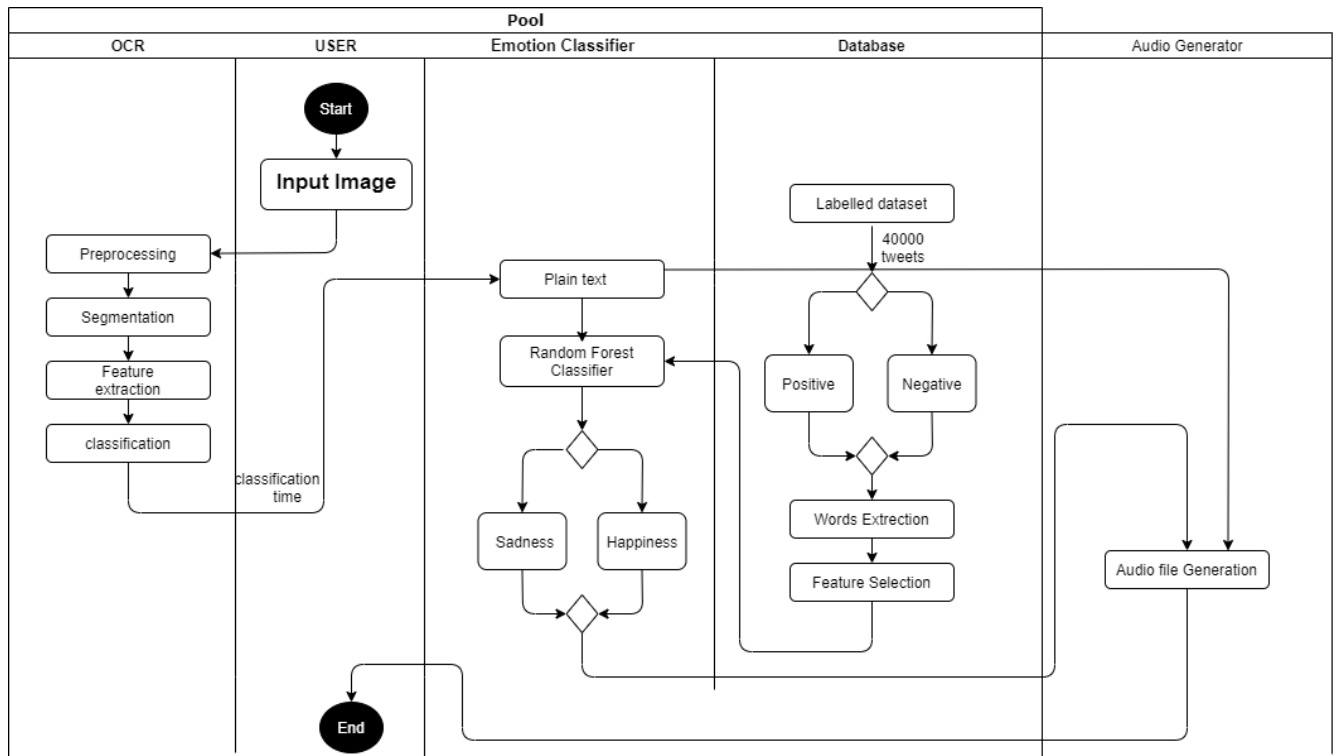


Fig 4.1 System Work Flow Diagram

4.5 SEQUENCE DIAGRAM

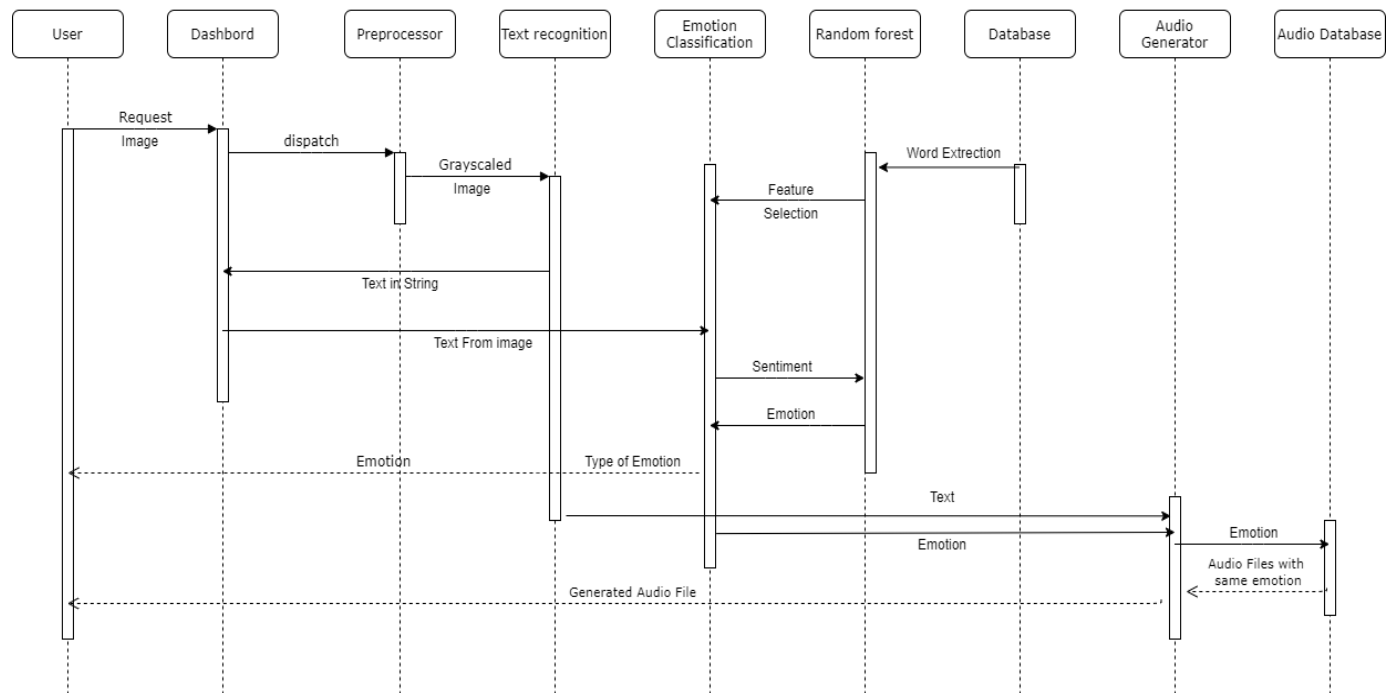


Fig 4.2 Sequence Diagram

4.6 STATE CHART DIAGRAM

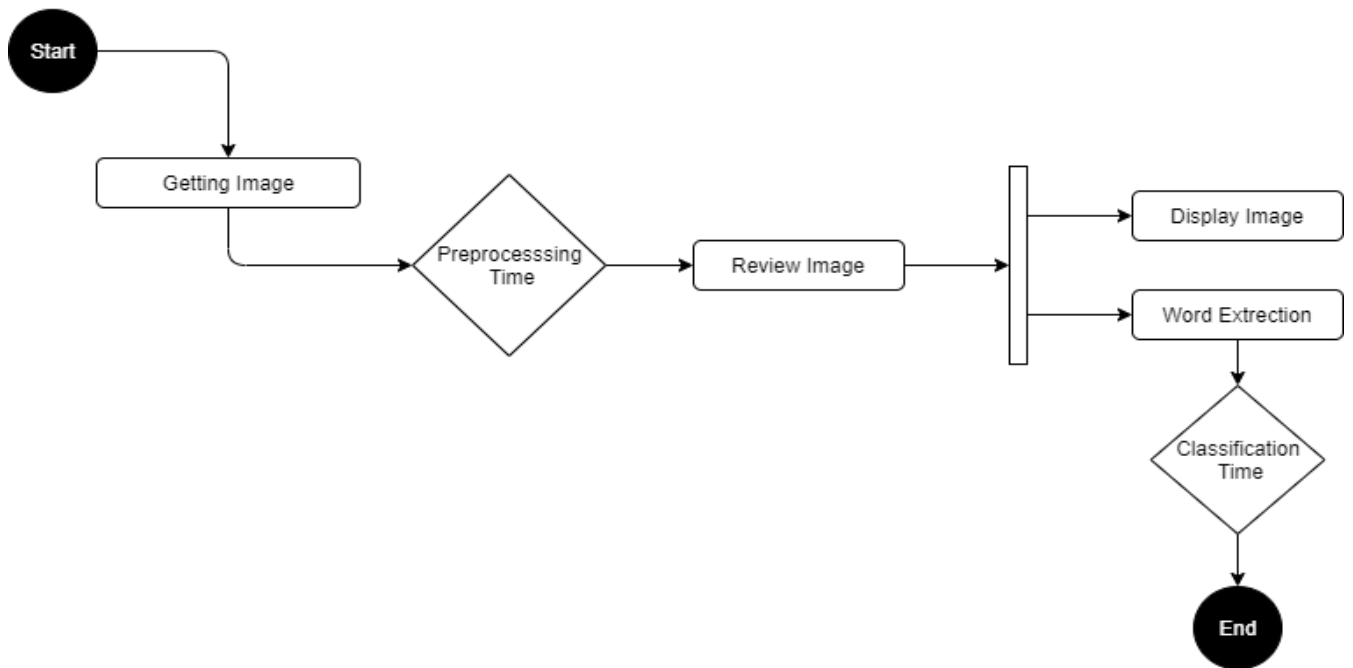


Fig 4.3 State Chart Diagram

4.7 ACTIVITY DIAGRAM

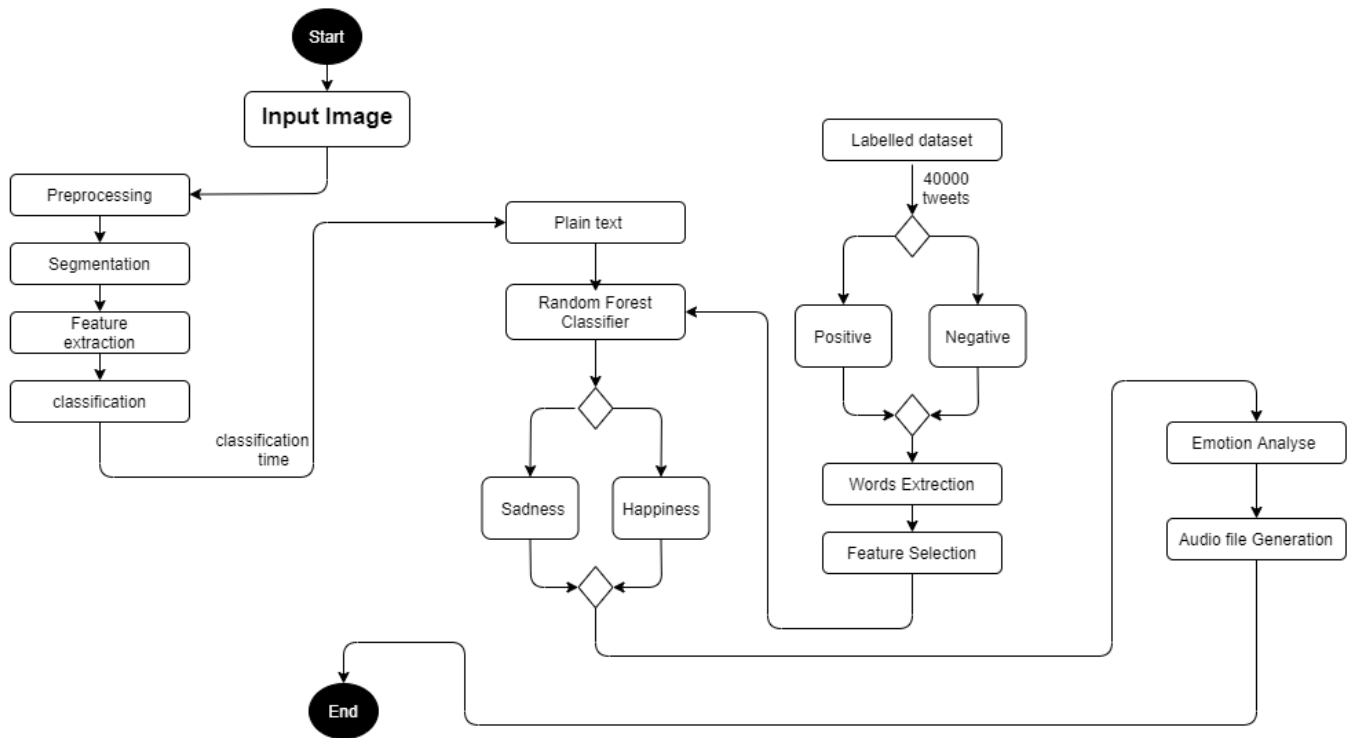


Fig 4.4 Activity Diagram

CHAPTER 5: SYSTEM DESIGN

5.1 DATA DICTIONARY

The database which we have used in our project is a text emotion database. We used this database to classify emotion from the text. Information of database is following.

Dataset Size: 4.19 MB

Dataset Tags: crowdsourced, social media, feelings, emotions.

Dataset Dictionary: 1 file, 4 columns, 0 tables.

Dataset Link: <https://data.world/crowdflower/sentiment-analysis-in-text>

Table 5.1 Dataset Summary

Column Name	Type
tweet_id	integer
sentiment	string
Author	string
content	string

Dataset Summary:

In a variation on the popular task of sentiment analysis, this dataset contains labels for the emotional content (such as happiness, sadness, and anger) of texts. Hundreds to thousands of examples across 13 labels. A subset of this data is used in an experiment we uploaded to Microsoft's Cortana Intelligence Gallery. Added: July 15, 2016 by CrowdFlower | Data Rows: 40000

Here, in this dataset, they have 13 different emotions, but we considered only two emotions in our project, the first is happiness and the second is sadness. So, for these two emotions only, we did not require to train our model with all 40 thousand tweets. We used only 10,374 tweets which have these two emotions only, happiness and sadness.

5.2 SCREEN LAYOUT (FORMS AND REPORTS)

- **Optical Character Recognition**

We have performed Optical Character Recognition with multiple different techniques like OpenCV, Tesseract, TensorFlow.

- **OCR with OpenCV and Tesseract**

In this method, we have taken an image with noisy background in order to check accuracy of model. After taking an input image, this model initially performs image preprocessing and then text recognition. To complete the whole process, we have used OpenCV for image preprocessing and Tesseract for text recognition.

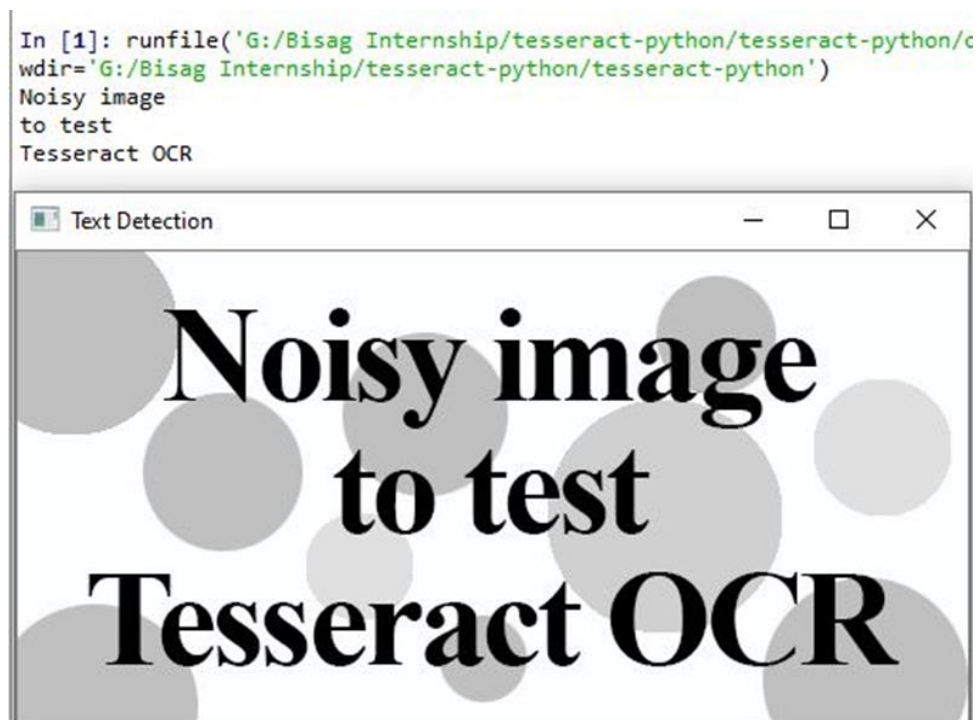


Fig 5.1 OCR output 1

Next, we have taken an image with large amount of text. In this image, we have tested same combination of image preprocessing and text recognition as we performed for previous image.

PREREQUISITES

In order to make the most of this, you will need to have a little bit of programming experience. All examples in this book are in the Python programming language. Familiarity with Python or other scripting languages is suggested, but not required.

You'll also need to know some basic mathematics. This book is hands-on and example driven: lots of examples and lots of code, so even if your math skills are not up to par, do not worry! The examples are very detailed and heavily documented to help you follow along.

In [4]:

```
In [4]: runfile('E:/Stuff Of Study/Internship/Internship 8th Sem/Projects/
tesseract-python/tesseract-python/ocr.py', wdir='E:/Stuff Of Study/
Internship/Internship 8th Sem/Projects/tesseract-python/tesseract-python')
PREREQUISITES
```

In order to make the most of this, you will need to have a little bit of programming experience. All examples in this book are in the Python programming language. Familiarity with Python or other scripting languages is suggested, but not required.

You'll also need to know some basic mathematics. This book is hands-on and example driven: lots of examples and lots of code, so even if your math skills are not up to par, do not worry! The examples are very detailed and heavily documented to help you follow along.

Fig 5.2 OCR output 2

Afterwards, we have chosen couple of real time image to check that same pair of image preprocessing and text recognition is successfully working or not. The results are revealed in the following pictures.



Fig 5.3 OCR output 3



Fig 5.4 OCR output 4

- **OCR with OpenCV and TensorFlow**

Then, we have changed our code to recognize handwritten text. In that for image preprocessing, we have used the same library named OpenCV, but for text recognition, we had changed to TensorFlow from tesseract. In which, TensorFlow used to create CNN model, which is used further for text recognition.

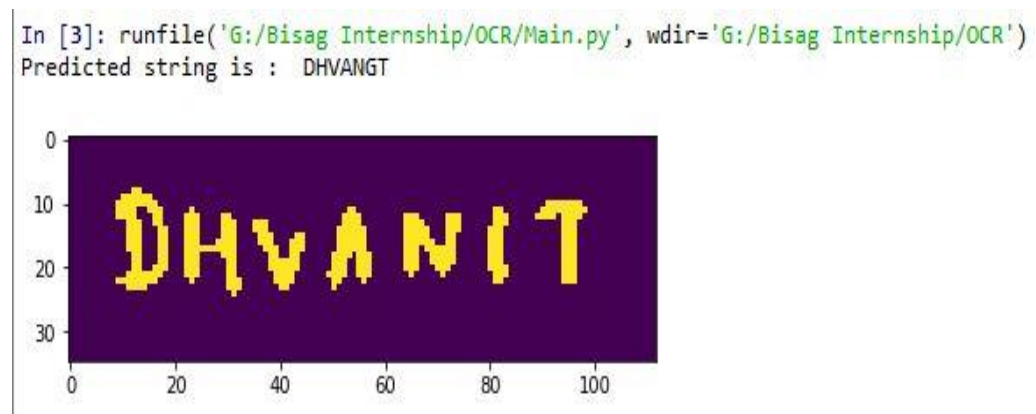


Fig 5.5 OCR output 5

- **Classification of Emotion from Detected Text**

The next phase of our application is to classify one emotion from detected text from an inputted image. So, for that we have tested some sentences in twitter sentimental analysis code. Currently, we have tested only two sentiments: happiness and sadness. The results for each sentiment are as following.

```
[48] text = 'Things are looking great. It was such a good day'
      tweets = pd.DataFrame([text])

[49] # Doing some preprocessing on these tweets as done before
      tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
      from nltk.corpus import stopwords
      stop = stopwords.words('english')
      tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
      from textblob import Word
      tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
      # Extracting Count Vectors feature from our tweets
      tweet_count = count_vect.transform(tweets[0])
      #Predicting the emotion of the tweet using our already trained linear SVM
      tweet_pred = lsvm.predict(tweet_count)
      print(tweet_pred)

[0]
```

```
if (tweet_pred == [0]):
    emotion = "happiness"
else:
    emotion = "sadness"

print(emotion)

happiness
```

Fig 5.6 Emotion Classification output 1

Prediction

```
text = "I am very happy today."
tweets = pd.DataFrame([text])

# Doing some preprocessing on these tweets as done before
tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
from nltk.corpus import stopwords
stop = stopwords.words('english')
tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
from textblob import Word
tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
# Extracting Count Vectors feature from our tweets
tweet_count = count_vect.transform(tweets[0])
#Predicting the emotion of the tweet using our already trained linear SVM
tweet_pred = lsvm.predict(tweet_count)
print(tweet_pred)

[0]
```

```
if (tweet_pred == [1]):
    emotion = "sadness"
else:
    emotion = "happiness"
print(emotion)

happiness
```

Fig 5.7 Emotion Classification output 2

```
[51] text = 'This is quite depressing. I am filled with sorrow'
      tweets = pd.DataFrame([text])

[52] # Doing some preprocessing on these tweets as done before
      tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
      from nltk.corpus import stopwords
      stop = stopwords.words('english')
      tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
      from textblob import Word
      tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
      # Extracting Count Vectors feature from our tweets
      tweet_count = count_vect.transform(tweets[0])
      #Predicting the emotion of the tweet using our already trained linear SVM
      tweet_pred = lsvm.predict(tweet_count)
      print(tweet_pred)

[1]

if (tweet_pred == [0]):
    emotion = "happiness"
else:
    emotion = "sadness"

print(emotion)

sadness
```

Fig 5.8 Emotion Classification output 3

Prediction

```
text = "More than 800 people died in Spain over the last 24 hours due to corona virus."
tweets = pd.DataFrame([text])

# Doing some preprocessing on these tweets as done before
tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
from nltk.corpus import stopwords
stop = stopwords.words('english')
tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
from textblob import Word
tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
# Extracting Count Vectors feature from our tweets
tweet_count = count_vect.transform(tweets[0])
#Predicting the emotion of the tweet using our already trained linear SVM
tweet_pred = lsvm.predict(tweet_count)
print(tweet_pred)

[1]

if (tweet_pred == [1]):
    emotion = "sadness"
else:
    emotion = "happiness"
print(emotion)

sadness
```

Fig 5.9 Emotion Classification output 4

- **Generate Output Speech with Classified Emotion**

The last phase of our project will be to generate output speech with the sentiment which is classified by sentimental analysis. But we have not successfully covered this part yet. We have completed our coding part, but we have not gotten success yet.

```

File "<ipython-input-1-12b7e5afc0a4>", line 1, in <module>
    runfile('G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master/inference.py',
wdir='G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master')

File "C:\Users\Nishit\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 827,
in runfile
    execfile(filename, namespace)

File "C:\Users\Nishit\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 110,
in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

File "G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master/inference.py", line 38, in
<module>
    denoiser = Denoiser(waveglow)

File "G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master/denoiser.py", line 30, in
__init__
    bias_audio = waveglow.infer(mel_input, sigma=0.0).float()

File "G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master/glow.py", line 276, in
infer
    output = self.WN[k]((audio_0, spect))

File "C:\Users\Nishit\Anaconda3\lib\site-packages\torch\nn\modules\module.py", line 532, in __call__
    result = self.forward(*input, **kwargs)

File "G:/Bisag Internship/Generate-Audio-From-Emotions-master/tacotron2-master/glow.py", line 159, in
forward
    spect = self.cond_layer(spect)

File "C:\Users\Nishit\Anaconda3\lib\site-packages\torch\nn\modules\module.py", line 576, in __getattr__
    type(self).__name__, name))

AttributeError: 'WN' object has no attribute 'cond_layer'

```

Fig 5.10 Text to Speech output

5.3 METHOD PSEUDO CODE

- INPUT: An Image with Text
- OUTPUT: The audio speaks the text within the image with emotion.

Code: Optical Character Recognition with OpenCV and Tesseract

```
image = get user input image
## Perform Some Image Preprocessing Techniques
image = convert image into grayscale
if
    image = apply thresholding
else
    image = apply median blurring
file = store grayscale image to disk as temporary file
Load image as PIL file
text = apply OCR with tesseract
print(text)
```

Code: Optical Character Recognition with OpenCV and TensorFlow

```
model = load the pretrained model
image = get user input image
## Perform Some Image Preprocessing Techniques
image = convert into grayscale
image = perform gaussian blurring on gray scaled image
image = perform thresholding
image = perform contours for text detection
## Bound the detected text and define region of interest (ROI)
prediction = send the ROI data into the model
text = get text from prediction
print(text)
```

Code: Classifying Emotion from The Text

```
## Import libraries and load the dataset
## We have used only happiness and sadness as emotion so we have dropped others
data = data.drop()
data = make all data in lower case
data = remove all punctuations
data = remove stop words
data = convert the words in root form
pattern = correct repetition
data = find unique words and remove rest
## Perform feature extraction techniques
encode = perform one hot encoding
data = split data in training and testing
tfidf = extract tf-idf parameters
count = transform words in array and get the number of time word appears
model = train the model
text = get the text for prediction
text = preprocess the before send it to the model
prediction = predict the emotion
print(prediction)
```

Code: Generate Final Output Audio with Emotion

```
## import libraries
## complete the setup process
## load the trained model
model = load_model(hparams)
## load waveglow for audio synthesis
waveglow = torch.load(waveglow_path)
text = get the text for input
text = generate the melspectrogram
```

```
## synthesis audio from spectrogram using waveglow  
audio = waveglow.infer(mel_outputs_postnet, sigma=0.666)  
audio = remove waveglow bias and get the final audio
```


CHAPTER 6: SYSTEM IMPLEMENTATION & TESTING

6.1 CODING STANDARDS

- In Python names are used as identifiers of functions, classes, variables, etc. Identifiers must start with a Letter (A-Z) or an underscore (“_”), followed by more letters or numbers.
- Python does not allow characters such as @, \$, and % within identifier names.
- Python comments are started with a hash (#) sign. The hash sign can be used at the start of a line, followed by a single line comment.
- Python does not use braces to denote block statement. Block statements are created using whitespace to the left of the lines within the block.
- Indent standard nested blocks four spaces. Indent the overview comments of a procedure one space.
- Indent the highest-level statements that follow the overview comments two spaces, with each nested block indented an additional two spaces.

6.2 TESTING METHODS

- The main objective of testing is to check that the developed project’s functionalities are satisfying the all system requirements or not.
- Before the testing of product, all methods and strategy need to plan accordingly. This planning will provide the environment and the platform for testing the project.
- For testing of this project, we have chosen black box testing because of following reasons:
 - Focusing on Software Functionality
 - Simplicity
 - Conserves Resources

Advantages:

- Black box tests are always executed from a user's point of view since it would help in exposing discrepancies significantly.
- Test cases related to black box are designed by testers as soon as the specifications are in the completed stage.
- Black box testing is all regarding the functional requirements of the system and it is executed by software testers.
- Software testers distinguish between the actual outputs with the expected outputs and check for similarities, if any noticeable differences are found, they are fixed and re-tested.

6.3 TEST SUITES DESIGN

Choose a test suite if each test case represents a component of a scenario, such as those elements that simulate a transaction completion. For instance, a test suite might contain three test cases, each with a separate test script:

- Test Case 1: Optical Character Recognition
- Test Case 2: Classify an emotion from the detected text
- Test Case 3: Generate output audio with classified emotion

Test suites can identify gaps in a testing effort where the successful completion of one test case must occur before you begin the next test case. For instance, you cannot add new products to a shopping cart before you successfully log in to the application. When you run a test suite in sequential mode, you can choose to stop the suite execution if a single test case does not pass. Stopping the execution is useful if running a test case in a test suite depends on the success of previous test cases.

Test suites are also useful for the following types of tests:

- **Build verification tests:** A collection of test cases that perform a basic validation of most the functional areas in the product. The tests are executed after each product build and before the build is promoted for use by a larger audience.
- **Smoke tests:** A collection of test cases that ensure basic product functionality. Typically, smoke tests are the first level of testing that is performed after changes are made to the system under test.
- **End-to-End integration tests:** A collection of test cases that cross product boundaries and ensure that the integration points between products are exercised and validated.
- **Functional verification tests:** A collection of test cases that focus on a specific product function. Executing this type of test with a test suite ensures that several aspects of a specific feature are tested.
- **Regression tests:** A collection of test cases that are used to make a regression pass over functional product areas.

6.4 TEST CASES

Test Case 1: Optical Character Recognition

Table 6.1 OCR Test Case 1

Test Cases	Expected Result	Actual Result	Component	Priority	Status
Load the input image	Load image successfully	Image loaded successfully	Functional	Critical	PASS
Convert image into grayscale	Image converted into grayscale	Image converted into grayscale	Functional	Major	PASS
Apply either thresholding or median blurring	Apply required processing	Successfully applied required processing	Functional	Major	PASS

Load image into disk	Load Image	Image loaded	Functional	Critical	PASS
Load image as PIL file	Load as PIL file	Loaded as PIL file	Functional	Critical	PASS
Apply OCR and print the text	Print recognized text	Recognized text printed	Functional	Critical	PASS

Table 6.2 OCR Test case 2

Test Cases	Expected Result	Actual Result	Component	Priority	Status
Import libraries	Successfully import libraries	Successfully imported libraries	Functional	Critical	PASS
Load previously train model	Successfully load model	Model loaded successfully	Functional	Critical	PASS
Input image	Successfully input image	Image inputted successfully	Functional	Critical	PASS
Perform Gaussian blurring	Successfully perform process	Process performed successfully	Functional	Major	PASS
Perform Thresholding	Successfully perform thresholding	Thresholding performed successfully	Functional	Major	PASS
Bound detected text	Successfully bound text	Text bounded successfully	Functional	Major	PASS
Get ROI	Successfully get ROI	Got ROI successfully	Functional	Major	PASS
Pass data into model	Successfully pass data	Data passed successfully	Functional	Critical	PASS
Get the predicted text	Successfully get text	Got the text successfully	Functional	Critical	PASS

Test Case 2: Classify an emotion from the detected text

Table 6.1 Emotion Classification Test Case

Test Cases	Expected Result	Actual Result	Component	Priority	Status
Import or download required libraries	Successfully Import or Download required Libraries	Successfully imported or downloaded required libraries	Functional	Critical	PASS
Load Dataset	Successfully load dataset	Dataset successfully loaded	Functional	Critical	PASS
Remove noisy data	Successfully remove noisy data	Successfully removed noisy data	Functional	Major	PASS
Make all data in lower case	Successfully make all data in lower case	Successfully made all data in lower case	Functional	Major	PASS
Remove punctuations from data	Successfully remove punctuations	Successfully removed punctuations	Functional	Major	PASS
Remove stop words	Successfully remove stop words	Successfully removed stop words	Functional	Major	PASS
Convert words into root form	Successfully convert into root form	Successfully converted into root form	Functional	Major	PASS
Correct letter repetition	Successfully correct repetition	Repetition successfully corrected	Functional	Major	PASS
Find unique data	Successfully find unique data	Unique data successfully found	Functional	Critical	PASS
Perform Encoding	Successfully perform encoding	Encoding performed successfully	Functional	Major	PASS
Split data for training and testing	Successfully split data	Data splinted successfully	Functional	Critical	PASS

Train Model	Successfully train model	Model trained successfully	Functional	Critical	PASS
Input text to fine emotion	Input Text successfully	Text inputted successfully	Functional	Critical	PASS
Perform Preprocessing	Successfully perform preprocessing	Preprocessing performed successfully	Functional	Critical	PASS
Predict Emotion	Successfully predict emotion	Emotion predicted successfully	Functional	Critical	PASS

Test Case 3: Generate output audio with classified emotion

Table 6.2 Generate Output Speech Test Case

Test Cases	Expected Result	Actual Result	Component	Priority	Status
Import libraries	Successfully import libraries	Successfully imported libraries	Functional	Critical	PASS
Setup environment	Successfully setup required environment	Successfully setup environment	Functional	Critical	PASS
Load model	Successfully load model	CUDA Error	Functional	Critical	FAIL
Load waveglow	Successfully load waveglow	Pending	Functional	Major	FAIL
Input the text for audio synthesis	Input text	Pending	Functional	Critical	FAIL
Generate mel spectrogram	Successfully generate spectrogram	Pending	Functional	Major	FAIL
Synthesis audio from spectrogram	Successfully synthesis audio	Pending	Functional	Major	FAIL
Get the final audio	Successfully get the audio	Pending	Functional	Critical	FAIL

CHAPTER 7: FUTURE ENHANCEMENT

7.1 FUTURE ENHANCEMENT

- Our application can recognize text of only one language, English. So, to expand features, in future, we could add more language support in text recognition.
- To add more features, we could add multi language speech support, in result of this, we would get output speech in different language.
- Another feature to add in upcoming time is language translation. In which, we would input image with text of any language and after translation, we would get output speech in any language we want.
- In output speech, we have considered only two emotions: happiness and sadness. To make our application more advance, we would add more emotions like neutral, anger, worry, surprise etc.
- Increase accuracy of model to get perfect output.

CHAPTER 8: CONCLUSION

8.1 SELF-ANALYSIS OF PROJECT VIABILITIES

- We have completed this project with primary functionalities as specified earlier, but then again there is a lot more than this which could be done. The main objective of project, which we have successfully covered, is to generate a human like voice with emotion from the text which is extracted from the image. It was a formidable challenge to construct a fresh application with given requirements, as there was not any reference application to get a general idea about the project.

8.2 PROBLEMS ENCOUNTERED AND POSSIBLE SOLUTION

- We were getting lower accuracy in detecting text from the image. Originally, we were not able to get every text from the image. So, as part of solution, we have used OpenCV for image preprocessing, which helped us to get higher accuracy than we previously got.
- After detecting the text with high accuracy, we were getting lower accuracy in text recognition also. To overcome with this problem we have used tesseract, with that we got satisfying accuracy.
- Next problem we had faced at the classification of emotion from the text, but after using random forest classifier, we got success in classifying emotion.
- The major problem, we have faced, is loading our waveglow in our final code of generating audio with classified emotion. We are continuously getting CUDA error and yet we are not able to find the solution.

8.3 SUMMARY OF PROJECT WORK

- We have completed our project with the proper and suitable approach of software engineering and system analysis and design, as well as the outcome of our project, is abiding the rules and requirements decided before the development of project. The concept of our project is newfangled, so the development of the project with instructed features was challenging.

REFERENCES

- Data.world. Sentiment Analysis in Text. (Dataset in Crowdfunder in 2017) from <https://data.world/crowdfunder/sentiment-analysis-in-text#>
- Flowchart Maker and Online Diagram Software. Drwa.io – Diagrams.net.
- Generate audio from Emotions (Jan 2020). Github Repository. <https://github.com/ChandhiniG/Generate-Audio-From-Emotions>
- IBM Engineering – Help. (Test cases and test suites) from jazz.net
- Javatpoint. Agile process model from <https://www.javatpoint.com/software-engineering-agile-model>
- Online Diagram Software & Visual Solution | LucidChart from Lucidchart.com
- Optical Character Recognition (2018). Github Repository. <https://github.com/dhaval1212/Optical-Character-Recognition>
- pyimagesearch (2018). OpenCV OCR and text recognition with Tesseract. Retrieved by Adrian Rosebrock on 17 september, 2018 from <https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/>
- pyimagesearch (2017). Using Tesseract OCR with Python. Retrieved by Adrian Rosebrock on 10 July, 2017 from <https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/>

- ResearchGate(2016). A Detailed Analysis of Optical Character Recognition Technology by Karez Hamad and Mehmet kaya on December 2016 from <https://www.researchgate.net/publication/311851325>

ORIGINALITY REPORT

29%

SIMILARITY INDEX

28%

INTERNET SOURCES

5%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1

www.slideshare.net

Internet Source

7%

2

www.javatpoint.com

Internet Source

4%

3

www.ijert.org

Internet Source

3%

4

idm.net.au

Internet Source

3%

5

www.ques10.com

Internet Source

2%

6

developer.rhino3d.com

Internet Source

2%

7

www.educba.com

Internet Source

1%

8

www.crowdfunder.com

Internet Source

1%

9

ccsindian.in

Internet Source

1%

10	www.ukessays.com Internet Source	1 %
11	www.j2eeonlinetraining.net Internet Source	1 %
12	www.scribd.com Internet Source	1 %
13	www.kyb.tuebingen.mpg.de Internet Source	<1 %
14	Zishan Ahmad, Raghav Jindal, Asif Ekbal, Pushpak Bhattacharyya. "Borrow from rich cousin: transfer learning for emotion detection using cross lingual embedding", Expert Systems with Applications, 2020 Publication	<1 %
15	publikationen.uni-tuebingen.de Internet Source	<1 %
16	www.essaysauce.com Internet Source	<1 %
17	documents.mx Internet Source	<1 %
18	szczecincafe.com Internet Source	<1 %
19	library.atmiya.net:8080 Internet Source	<1 %

20

www.dcs.kcl.ac.uk

Internet Source

<1 %

21

dspace.lib.cranfield.ac.uk

Internet Source

<1 %

22

Priyanka Patel, Amit Thakkar. "The upsurge of deep learning for computer vision applications", International Journal of Electrical and Computer Engineering (IJECE), 2020

Publication

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off