

SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

**Improving Human-Computer Interaction
with Machine Emotion Intelligence using
NAO Robot**

**SUBMITTED TOWARDS THE
FULFILLMENT OF THE REQUIREMENTS OF**

**BACHELOR OF ENGINEERING (Computer
Engineering)**

BY

Manas Kale (B120024271)
Rohit Patankar (B120024292)
Saurabh Shiroadkar (B120024325)
Shubham Puneekar (B120024304)

Under The Guidance of

Dr. V.Y. Kulkarni



**Department of Computer Engineering
MAEER's MAHARASHTRA INSTITUTE OF TECHNOLOGY
Kothrud, Pune 411 038
2016-2017**



Maharashtra Institute of Technology, Pune
DEPARTMENT OF COMPUTER ENGINEERING

C E R T I F I C A T E

This is to certify that the Project Entitled

**Improving Human-Computer Interaction with
Machine Emotion Intelligence using NAO Robot**

Submitted by:

Manas Kale (B120024271)
Rohit Patankar (B120024292)
Saurabh Shirodkar (B120024325)
Shubham Puneekar (B120024304)

is a bonafide work carried out by students under the supervision of Dr. V.Y. Kulkarni and it is submitted towards the fulfillment of the requirement of Bachelor of Engineering (Computer Engineering).

Prof. Dr. V.Y. Kulkarni
(Internal Guide and Head of Computer Engineering Department)
Place: Pune Date: 03/05/2018

PROJECT APPROVAL SHEET

Improving Human-Computer Interaction with Machine Emotion
Intelligence using NAO Robot

Is successfully completed by

Manas Kale (B120024271)
Rohit Patankar (B120024292)
Saurabh Shiroadkar (B120024325)
Shubham Punekar (B120024304)

at

DEPARTMENT OF COMPUTER ENGINEERING

Maharashtra Institute of Technology, Pune

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2017-2018

Dr. V.Y. Kulkarni (Dept. of Computer Engg.)

Acknowledgements

It gives us great pleasure in presenting the project report on
**”Improving Human-Computer Interaction with Machine Emotion
Intelligence using NAO Robot”**.

We would like to take this opportunity to thank our internal guide and H.O.D. **Dr. V.Y. Kulkarni** for giving us all the help and guidance we needed. Her valuable suggestions were extremely helpful. We would also like to express our sincere appreciation to staff of department of Computer Engineering, Maharashtra Institute of Technology Pune, for their extended help and suggestions at every stage. In the end our special thanks to our college support staff for providing various resources such as laboratory whenever we wanted with all needed software platforms, continuous internet connection, and air conditioning for our project work.

Contents

1	Synopsis	1
1.1	Project Statement	1
1.2	Scope of the Problem	1
1.3	Goals and Objectives	2
1.4	Relevant mathematics associated with the Project	3
2	Technical Keywords	9
2.1	Area of Project	9
2.2	Technical Keywords	9
3	Introduction	11
3.1	Project Idea	11
3.2	Motivation of the Project	11
3.3	Literature Survey	12
4	Problem Definition and scope	19
4.1	Problem Statement	19
4.2	Statement of scope	19
4.3	Major Constraints	20
4.4	Outcome	20
4.5	Applications	20
4.6	Software Resources Required	21
4.7	Hardware Resources Required	21
4.8	Project Resources	21
4.9	Risk Management w.r.t. NP Hard analysis	23
4.9.1	Risk Identification	23
4.9.2	Overview of Risk Mitigation, Monitoring, Management	25
5	Software requirement specification	28
5.1	Introduction	28
5.1.1	Purpose and Scope of Document	28

5.1.2	Intended Audience and Reading Suggestion	29
5.2	Usage Scenario	29
5.2.1	Use-cases	29
5.2.2	Use Case View	31
5.3	Functional Model and Description	32
5.3.1	Data Flow Diagram	32
5.3.2	Class Diagram	34
5.3.3	Activity Diagram	34
5.3.4	State Diagram	36
5.3.5	Deployment Diagram	37
5.3.6	Non Functional Requirements	37
5.4	Architectural Design	39
6	Project Implementation	41
6.1	Methodologies/Algorithm Details	41
6.1.1	Algorithm/Pseudo Code	41
6.2	Reports for classifier training process	47
7	Software Testing	61
7.1	Type of Testing Used	61
8	Results	67
8.1	Outputs	67
8.1.1	Casting audition monologue by Breeze Woodson	67
8.1.2	Audition monologue "Getting out" by Marsha Norman	67
8.1.3	Reactions - Unfair Mario gameplay	68
8.1.4	Macbeth monologue by Andrew Serkis	68
8.1.5	Testing accuracy for real-time analysis	68
8.2	Screenshots	69
9	Deployment and Maintenance	72
9.1	INSTALLATION	72
10	Conclusion and future scope	76
Annexure A Laboratory assignments on Project Analysis of Algorithmic Design		78
Annexure B Term-II Project Laboratory Assignments		88

Annexure C	Information of Project Group Members	94
C.1	94
C.2	95
C.3	96
C.4	97

List of Figures

5.1	Use case diagram	31
5.2	Dataflow diagram	33
5.3	Class Diagram	35
5.4	Activity Diagram	36
5.5	State Diagram	37
5.6	Deployment Diagram	38
5.7	Architecture diagram	40
6.1	Facial Landmarks for detection	43
6.2	Facial Landmarks for detection Example	44
6.3	45
6.4	Tone Utterances before combining labels	48
6.5	Tone Utterances after combining labels	49
6.6	Learning Curve	51
6.7	Confusion Matrix	51
6.8	Visual Corpus Distribution	53
6.9	Learning Curve	56
6.10	Confusion Matrix	56
6.11	Accuracy with change in input size	58
6.12	Precision and Recall with change in input size	58
6.13	Speech Module Confusion Matrix	59
6.14	Speech Module Confusion Matrix	60
7.1	Video Module Unit Test	63
8.1	Project Execution	69
8.2	Neutral Emotion Recognition in Real Time	70
8.3	Anger Emotion Recognition in Real Time	70
8.4	Happy Emotion Recognition in Real Time	71
9.1	Project Structure	73
A.1	Idea Matrix	78

A.2	Tone Analysis	80
A.3	Speech Analysis	82
A.4	Facial Analysis	83
A.5	Response Generation	84
A.6	Parallelism in the project	86

List of Tables

4.1	Hardware Requirments	22
4.2	Risk Table	23
4.3	Risk Probability definitions	24
4.4	Risk Impact definitions	24
5.1	Use Cases	30
6.1	Risk Impact definitions	57

Abstract

In our project, we have focused on inculcating machine emotion intelligence using the NAO robot. Emotion detection is a challenging problem partly because it is difficult to determine which features of the interaction might be relevant to the task, and partly because the emotive states are overlapping and not mutually exclusive. Emotional state of a person at any given time is not categorical, but a value from a wide spectrum of emotions.

Thus, we have pre-defined a limited set of emotions and focused on an ensemble approach by considering the interacting human's voice, spoken content as well as facial cues to detect the emotional state. We have used a pipeline of deep neural network for tone analysis, an SVM for facial feature analysis and a naive bayes model for analysing the spoken text. By using a weighted function, an output is calculated from probability distributions for emotion labels predicted by all 3 modules. The robot will accept a multimodal input periodically and output the associated recognized emotion label on the user interface along with graphs to show the changing emotions with respect to every sample taken.

Chapter 1

Synopsis

1.1 Project Statement

Improve Human Computer Interaction with Machine Emotional Intelligence using Nao Robot to:

- recognize subjects based on their facial features and voice
- analyse the facial features, speech text and the tone of speech to detect emotions
- generate an emotive score based on a weighted scores from former analyses

1.2 Scope of the Problem

- NAO robot will automatically and periodically analyze voice samples and facial cues in order to detect the emotional state of the person interacting with the robot.
- Specified number of frames per second will be analysed for facial cues.
- Audio segments will be analysed via tone for emotion detection.
- Speech text extracted from the audio segments will be aggregated and analysed for emotion.
- The robot will not be able to detect every single complex emotion, but will be limited to a subset of generalized emotions.

1.3 Goals and Objectives

- Knowing the emotional state of an individual can be crucial in determining what action is to be taken as a response.
- Recognizing the affective state of a human can be difficult for humans as well as computer systems. Many features can be considered such as voice samples, facial cues or even text written by the person to identify the emotional state of the individual.
- The major focus of the project is improving human-machine interaction using the NAO robot.
- The robot will accept the input from the person periodically in the form of speech samples, comprising of voice and text as well as facial cues and will interpret the current emotional state of the person.
- Although our main focus is on humanizing the NAO robot, there are myriad of other uses that can be achieved; some of which are:
 - Development of an affect-aware city
 - Add security layer at public venues to detect malicious intent and deal with hostage situation effectively
 - Measure response and ratings in focus groups (consumer response to commercials etc)
 - Wearables that help autistics discern emotion etc.

1.4 Relevant mathematics associated with the Project

System Description:

The System is divided into three parts, each of which handles a mode of emotion recognition. Each part has its own specification of inputs. However, the system as a whole has output in terms of final output emotion label as well as associated graphs. The system takes three types of inputs(audio, face and text samples) which are given to the associated part for emotion recognition.

Consider System as S:

$$S = \{ I, O, F, SC, FC \}$$

where,

- I = Input
- O = Output
- F = Functions
- SC = Success Cases
- FC = Failure Cases

- **I:** Multimodal input captured using Camera and Microphone/ stored file

$I = \{ \text{Audio and Video sampling is customizable. However, the default sampling is at 1 second and in case of significant input, 5 second utterances are captured} \}$

- **Output:** Distinct emotion label from the pre-defined set of emotions
emotions = {Neutral, Sadness/Fear, Frustration/Anger/Disgust, Happiness/Excitement/Surprise}

- F : { F1, F2, F3}

– F1 = **Data Preprocessing**

* **Tone Module**

· **Functions =**

{ **F1:** Determine threshold with amplitudes for silence

$$threshold = \theta | \theta = \max S_0, \dots, S_k \quad (1.1)$$

where S_i = Sampled integer during base-line seconds used for measuring threshold

F2: Generate MFCCs for each frame $z(m)$

F3: Generate MFCC matrix for segments. 2m+1 frames of 25 ms each are stacked to produce one segment $x(m)$ of 265 ms each using sliding window of $w = 10$ ms

$$x(m) = [z(m - w), \dots, z(m), \dots, z(m + w)] \quad (1.2)$$

F4: Generate statistical features from probability distribution

$$(f_1)^k = \max_{S \in U} P_s(E_k) \quad (1.3)$$

where,

$P_s(E_k)$ denotes the probability for kth emotion for segment S in the utterance U }

· **SC =**

{

1. Appropriate threshold selected for ambient noise
2. Appropriate sliding window size selected to convert audio frames into segments.

}

· **FC =**

{

1. Improper threshold for ambient noise resulting in no audio utterance being generated

2. Improper window size selected resulting in improper MFCC matrix
- }

* **Video Module**

- **Functions** =
 - { **F1::** Optimize contrast by performing histogram equalization
 - F2:** Detect faces in the frame
 - F3:** Map 68 landmarks onto the detected face
 - F4:** Re-orient the face by taking the centroid of the detected landmarks and the normal
 - F5:** Generate features by calculating distances and angles of the landmarks from the centroid and normal respectively }
- **SC** =
 - {
 - 1. Faces detected in the frame, image having optimal contrast
 - 2. only one face utilized from multiple faces in the frame
 - }
- **FC** =
 - {
 - 1. Faces not detected in the frame, too dark/ bright/ blurred images for effective edge detection
 - 2. Face angle too oblique
 - }

* **Speech Module**

- **Function** =
 - { **transcribe()** : Takes the speech audio as input, and generates corresponding transcript.
 - transform()** - Transforms the transcript into a feature vector of a predefined format.
 - predict()** - Predicts the probabilities of the transcript belonging to each of the emotion labels. }

- **SC** =
 - {
 - 1. Audio stream is clear enough to generate transcript properly
 - 2. Stable internet connection available to access Google Speech to Text API
 - 3. Speech takes place in English language
 - 4. Feature vector is not too sparse
 - 5. Enough spoken words occur in the feature set to accurately discern between emotions
 - }
- **FC** =
 - {
 - 1. Too much noise in the audio stream to detect speech
 - 2. Unstable internet connection preventing consistent access to Google Speech to Text API
 - 3. Language other than English spoken
 - 4. Feature vector generated is too sparse to accurately discern between emotions
 - }

– F2 = **Training the Models**

* **Tone Module:**

- **Input :**
 - 1. Preprocessed data in the form of MFCC matrix of segments composed from frames.
 - 2. One feature vector contains 325 MFCCs (13 MFCCs for 25 frames each).
- **Output :** Probability distribution for each segment
- **Parameters :**
 - 1. 3 hidden layers of 250 neurons each (250,250,250)
 - 2. Input layer contains 325 neurons as per the dimensionality of the feature vector

3. Output layer is a softmax layer for probabilities of 4 emotion labels
4. Activation function used is ReLU
5. Learning algorithm is stochastic gradient descent
6. Objective function is cross entropy error

* **Video Module:**

· **Input :**

1. A feature vector of 136 dimensions (68 landmarks and 68 angles)

· **Output :**

1. Probability distribution of the 6 emotion labels

· **Parameters :**

1. SVM with a linear kernel is used

* **Speech Module:**

· **Input**

Dataset split into training set and testing(validation) set with k -fold cross-validation for assessing accuracy.

· **Output**

Trained classifier with Likelihood

· **Likelihood(evidence) calculation**

Conditional probabilities of attributes for class labels are calculated from the dataset, termed as evidence Z .

$$Z = p(x) = \sum_k p(C_k)p(x|C_k) \quad (1.4)$$

* **Classification stage**

This stage uses the likelihood or evidence calculated in training to classify a novel input.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \quad (1.5)$$

In text classification, the goal is to find the best class for the input text. The best class in Multinomial NB is the most

likely or maximum posteriori (MAP) class c_{map}

$$c_{map} = \underset{c \in C}{argmax} P(c|d) = \underset{c \in C}{argmax} P(c) \prod_{1 \leq k \leq n_d} P(tk|c) \quad (1.6)$$

In the above equation, many conditional probabilities are multiplied, and with a large enough vocabulary, raw multiplication will almost definitely result in an underflow. It is therefore better to perform the computation by adding logarithms of probabilities instead of multiplying probabilities. The class with highest log probability score is still the most probable; $\log(xy) = \log(x) + \log(y)$ and the logarithmic function is monotonic. Hence, the maximization that is actually done in our implementation of the Multinomial NB classifier is as follows:

$$c_{map} = \underset{c \in C}{argmax} [\log P(c) + \sum_{1 \leq k \leq n_d} \log P(t_k|c)] \quad (1.7)$$

• F3 = Emotion Recognition

- **Input:** Probability distributions for all emotion labels predicted by individual classifiers ($P_c(k)$) ; $c = 1,2,3$ for video, tone and speech modules respectively.
- **Output:** Weighted probability distribution for emotion labels from stacked classifiers
- **Processing:** Weighted average of probabilities from classifiers which are weighted by a time decaying factor

$$P_{(k)} = \sum_{c=1}^3 P_c(k)w_c \quad (1.8)$$

where,

$w_c = w - \Delta w$;

$\Delta w = d * t$;

d = decay factor ;

w = default maximum weight

Chapter 2

Technical Keywords

2.1 Area of Project

The area of our project lies in the intersection of three distinct areas which are Machine Learning, Computer Vision and Natural Language Processing.

2.2 Technical Keywords

- **Machine Learning:** Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.
- **Computer Vision:** Computer vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos.
- **Affective Computing:** Affective computing is the study and development of systems and devices that can recognize, interpret, process, and simulate human affects. It is an interdisciplinary field spanning computer science, psychology, and cognitive science.
- **Neural Network:** Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.
- **Deep Neural Network:** Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input

and output layers. DNNs can be used for modelling complex non-linear relationships.

- **SVM:** In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- **Naive Bayes Classifier:** naive Bayes classifier is a simple "probabilistic classifier" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. The naive Bayes classifier combines this probabilistic model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule.
- **Weighted Function:** A weight function is a mathematical device used when performing a sum, integral, or average to give some elements more "weight" or influence on the result than other elements in the same set. The result of this application of a weight function is a weighted sum or weighted average. Weight functions occur frequently in statistics and analysis, and are closely related to the concept of a measure. Weight functions can be employed in both discrete and continuous settings.

Chapter 3

Introduction

3.1 Project Idea

The basic idea behind our project was to combine the various forms of input that one usually analyses in order to determine a person's emotional state.

3.2 Motivation of the Project

- Majority of work done towards emotion detection is focused on a single mode i.e. audio/ video/ text. There is limited practical work done by considering multimodal inputs.
- In general literature available today, numerous features have been developed, however performance of classifiers is still limited as the emotional states cannot be accurately distinguished by a well-defined set of discriminating features.
- By considering multimodal input, the final output is completely influenced by the quality and accuracy of the input itself, thus in scenarios where one mode of input becomes unreliable, the output can be provided by another mode thereby maintaining the accuracy of identification of emotions.

3.3 Literature Survey

- **Jeong-Sik Park, Gil-Jin Jang: "Implementation of Voice emotion Recognition for Interaction with Mobile Agent , ACM 2014"**

The paper proposes a simple smartphone interface framework which consists of detection of human voice, extraction of emotional features and identification of an emotional state. Energy based approach for detection of human voice is selected in which if continues estimates of spectral energy of consecutive frames exceeds a pre-determined threshold, the region is regarded as starting point of voice signal. The pitch, log-energy, and Mel-Frequency Cepstral Coefficient (MFCC) are selected for extraction of emotional features which make a feature vector sequence. Acoustic features vectors extracted are analyzed and compared with patterns for each emotion type. Guassian Mixture Model is the classification algorithm used. This approach achieved 70.1% correctness within 1s response time. The future work suggested is applying proposed concepts for human-machine interaction in personal agent applications.

- **Yu Gu, Eric Postma, Hai-Xing Lin: "Vocal Emotion Recognition using Log-Gabor Filters , ACM 2015"**

The proposed work utilizes 2d Gabor filters in order to decompose the associated spectrogram in order to perform a spectro-temporal analysis of affective vocalizations. Instead of including all potentially relevant features which leads of dimensionality problem and subsequent degradation of performance, the work uses feature learning in which relevant features are automatically obtained from raw speech signals. However this leads to considerable computational resources. Hence, no. of features is kept to minimum. By performing analysis on local spectro-temporal structure, the spectrogram is treated as an image and standard image processing is implemented. Comparative evaluation of MFCC and LPCC features, untuned and tuned Gabor filters and all above combinations is done. SVM is used as a classifier. The confusion matrix for performance using tuned Gabor Filters provide a maximum of 91.6% accuracy whereas a combination of acoustic features and Gabor filter provides 93.5

- **Gloria Zen, Elisa Ricci, Nicu Sebe: "Unsupervised Domain Adaption for Personalized Facial Emotion Recognition, ACM**

2014”

A personalization approach is proposed in which only unlabeled target-specific data are required. A new method to represent the source sample distribution based on only Support Vectors of source classifiers is proposed. Regression framework is used to learn a mapping between a marginal distribution of the data points associated to a given person and the parameters of his/her personalized classifier which is represented by a set of Support Vectors of linear classifier in the source case and by all unlabeled data points in the target case.

- **Ahmed Mustafa Mahmoud, Wan Haslina Hassan: ”Determinism in Speech Pitch Relation to Emotions, ICIS 2009”**

A deterministic rule-based text-to-speech emotional synthesis approach is proposed to generate emotional speech using semitonic interval-driven rules. Emotional speech samples are analyzed and intervals are extracted using praat tool. Objective evaluation compares synthesized voice to natural voice and calculates difference as an error function by considering mean square error as a measure of similarity. New emotional states may be defined using same proposed approach. Algorithms that integrate two or more emotional states may be combined to generate a variety of complex emotions.

- **Nancy Semwal, Abhijeet Kumar, Sakthivel Narayanan: ”Automatic Speech Emotion Detection using Multi-Domain Acoustic Feature Selection and Classification, IEEE 2015”**

The proposed approach concentrated on determining emotions from speech signals. Various acoustic features such as energy, zero-crossing rate(ZCR), fundamental frequency, Mel Frequency Cepstral Coefficient are extracted for short term, overlapping frames derived from the input signal. A feature vector for every utterance is then constructed by analyzing mean, median, etc. over all frames. Sequential Backward Selection is used with K-fold cross validation to select a subset of useful features. Detection of emotions is done by classifying respective features from the full candidate feature vectors into classes, using either a pre-trained SVM or a Linear Discriminant Analysis classifier. Accuracy of 80% was obtained when tested on EmoDB dataset.

- **Lei Pang, Chong-Wah Ngo: ”Multimodal Learning with Deep Boltzmann Machine for Emotion Prediction, ACM 2015”**

In contrast to existing works which concentrate on either Audio, text or video, a joint density model is proposed over the space of multi-modal inputs with Deep Boltzmann Machine. The model is trained directly on user-generated Web videos without any labelling effort. Multiple layers of hidden units and multiple modalities make learning difficult, hence learning is split into 2 stages. First, each RBM component is pre trained using greedy layerwise strategy. Then, learnt parameters are used to initialize the parameters of all layers in DBM and then the multimodal DBM is trained to finetune different modalities in a unified way. A major factor is that the deep architecture enlightens the possibility of discovering highly non-linear relationships between low-level features across different modalities. A performance improvement of 7.7% in classification accuracy is observed.

- **Benjamin Guthier, Rajwa Alharthi, Rana Abaalkhail, Abdulmotaleb El Saddik: "Detection and Visualization of Emotions in an Affect-Aware City, ACM"**

In the proposed work, emotions are represented as four-dimensional vectors of pleasantness, arousal, dominance and unpredictability. In the training phase, emotion word hashtags in the messages are used as the ground-truth emotion contained in a message. A neural network is trained by using the presence of words, hashtags and emoticons in the message as features. During the live phase, these features are extracted from geo-tagged Twitter messages and given as input to neural-network. The detected emotions are aggregated over space and time and visualized on a map of the city.

- **Huaizu Jiang, Erik Learned-Miller: "Face Detection with Faster R-CNN"**

Most approaches to face detection are still based on the R-CNN framework, leading to limited accuracy and processing speed. In this paper, investigations regarding the application of Faster R-CNN which has demonstrated impressive results on various object detection benchmarks, to face detection have been made. By training a Faster R-CNN model on the large scale WIDER face dataset, state-of-the-art results on the WIDER test set as well as two other widely used face detection benchmarks, FDDB and the recently released IJB-A have been presented.

- **Wei Jang, Wei Wang: "Face Detection and Recognition for Home Service Robots with End-To-End Deep Neural Networks, IEEE 2017"**

This paper proposes an effective end-to-end face detection and recognition framework based on deep convolutional neural networks for home service robots. State-of-the-art region proposal based deep detection network has been combined with the deep face embedding network into an end-to-end system, so that the detection and recognition networks can share the same deep convolutional layers, enabling significant reduction of computation through sharing convolutional features. The detection network is robust to large occlusion, and scale, pose, and lighting variations. The recognition network does not require explicit face alignment, which enables an effective training strategy to generate a unified network. A practical robot system is also developed based on the proposed framework, where the system automatically asks for a minimum level of human supervision when needed, and no complicated region-level face annotation is required. Experiments are conducted over WIDER and LFW benchmarks, as well as a personalized dataset collected from an office setting, which demonstrate state-of-the-art performance of the system.

- **Rajesh K M, Naveenkumar M: "A Robust Method for face Recognition and Face Emotion Detection System using Support Vector Machines, IEEE 2016"**

This paper presents framework for real time face recognition and face emotion detection system based on facial features and their actions. The key elements of Face are considered for prediction of face emotions and the user. The variations in each facial feature are used to determine the different emotions of face. Machine learning algorithms are used for recognition and classification of different classes of face emotions by training of different set of images. In this context, by implementing herein algorithms would contribute in several areas of identification, psychological researches and many real world problems. The proposed algorithm is implemented using open source computer vision (OpenCV) and Machine learning with python.

- **Jie Shen, Ognjen Rudovic, Shiyang Cheng, Maja Pantic: "Sentiment Apprehension in Human-Robot Interaction with NAO"**

In this paper, the influence of sentiment apprehension by robots (i.e.,

robots ability to reason about the users attitudes such as judgment / liking) on the user engagement has been studied. Two versions of mimicry game are studied: in the first, NAO was solely mimicking facial expressions of the users, while in the second he was also providing a feedback based on the sentiment apprehension. A total of 32 participants (7 female, 25 male) were recruited for this experiment, and the results show that the participants in the second group spent more time interacting with the robot and played more rounds of the mimicry game. After experiencing both versions of the game, ratings given by the participants indicate (with 99% confidence) that the game with sentiment apprehension is more engaging than the baseline version.

- **Dario Bertero, Pascale Fung: "A First Look into Convolutional Neural Network for Speech Emotion Detection, IEEE 2017"**

A real-time Convolutional Neural Network model for speech emotion detection. Our model is trained from raw audio on a small dataset of TED talks speech data, manually annotated into three emotion classes: Angry, Happy and Sad. It achieves an average accuracy of 66.1%, 5% higher than a feature-based SVM baseline, with an evaluation time of few hundred milliseconds. An in-depth model visualization and analysis is also provided. How the neural network effectively activates during the speech sections of the waveform regardless of the emotion, ignoring the silence parts which do not contain information has also been shown. On the frequency domain the CNN filters distribute throughout all the spectrum range, with higher concentration around the average pitch range related to that emotion. Each filter also activates at multiple frequency intervals, presumably due to the additional contribution of amplitude-related feature learning.

- **Kun Han, Dong Yu, Ivan Tashev: "Speech Emotion Recognition and Extreme Learning Machine, INTERSPEECH 2014"**

Speech emotion recognition is a challenging problem partly because it is unclear what features are effective for the task. In this paper an approach is proposed to utilize deep neural networks (DNNs) to extract high level features from raw data and it is shown that they are effective for speech emotion recognition. First an emotion state probability distribution is produced for each speech segment using DNNs. Then utterance-level features from segment-level probability distribu-

tions are constructed. These utterancelevel features are then fed into an extreme learning machine (ELM), a special simple and efficient single-hidden-layer neural network, to identify utterance-level emotions. The experimental results demonstrate that the proposed approach effectively learns emotional information from low-level features and leads to 20% relative accuracy improvement compared to the state-of-the-art approaches.

- **Dan Duncan, Gautam Shine, Chris English: "Facial Emotion Recognition in Real Time"**

This paper proposes a convolutional neural network for classifying human emotions from dynamic facial expressions in real time. Transfer learning is used on the fully connected layers of an existing convolutional neural network which was pretrained for human emotion classification. A variety of datasets and homebrewed dataset is used to train the model. Overall training accuracy of 90.7% and test accuracy of 57.1% is achieved. A live video stream connected to a face detector feeds images to neural network. The network subsequently classifies an arbitrary number of faces per image simultaneously in real time. This paper essentially demonstrates the feasibility of implementing neural networks in real time to detect human emotions.

- **Lifeng Shang, Zhengdong Lu, Hang Li: "Neural Responding Machine for Short-Text Conversation"**

This paper proposes Neural Responding Machine (NRM), a neural network-based response generator for Short-Text Conversation. NRM takes the general encoderdecoder framework: it formalizes the generation of response as a decoding process based on the latent representation of the input text, while both encoding and decoding are realized with recurrent neural networks (RNN). The NRM is trained with a large amount of one-round conversation data collected from a microblogging service. Empirical study shows that NRM can generate grammatically correct and content-wise appropriate responses to over 75% of the input text, outperforming stateof- the-arts in the same setting, including retrieval-based and SMT-based models(Statistical Machine Translation or a generative model).

- **Joost Broekens: "Emotion and Reinforcement: Affective Facial Expressions Facilitate Robot Learning"**

Computer models can be used to investigate the role of emotion in learning. Here weThis paper presents EARL framework for the systematic study of the relation between emotion, adaptation and reinforcement learning (RL). EARL enables the study of communicated affect as reinforcement to the robot. In humans, emotions are crucial to learning. For example, a parent observing a child uses emotional expression to encourage or discourage specific behaviors. Emotional expression can therefore be a reinforcement signal to a child. We hypothesize that affective facial expressions facilitate robot learning, and compare a social setting with a non-social one to test this. The non-social setting consists of a simulated robot that learns to solve a typical RL task in a continuous grid-world environment. The social setting additionally consists of a human (parent) observing the simulated robot (child). The humans emotional expressions are analyzed in real time and converted to an additional reinforcement signal used by the robot; positive expressions result in reward, negative expressions in punishment. It is quantitatively shown that the social robot indeed learns to solve its task significantly faster than its non-social sibling. This paper concludes that this presents strong evidence for the potential benefit of affective communication with humans in the reinforcement learning loop.

Chapter 4

Problem Definition and scope

4.1 Problem Statement

- Our project addresses the lack of a system in practice today which considers more than one method of emotion recognition at the same time. Our goal is to design a system that considers audio, facial cues as well as the content spoken to recognize the emotion the speaker is experiencing. Making the system work in near-real time is also crucial to address the effectiveness of emotion recognition in a real-life scenario.

4.2 Statement of scope

- The system will accept three inputs. Out of the three inputs, two will be directly picked up by the camera and microphone on the robot.
- The third input will be given to the system after undergoing a Speech-to-Text conversion.
- The audio input will be approximately 5 seconds long in .wav format.
- The video frames will be captured every second.
- After outputs are displayed, the files will be deleted to address the storage space requirement.
- The system will give the output in the form of an emotion label after successful emotion recognition.
- However, this label will be from a pre-defined set of emotions.

- For successful Speech-to-Text conversion, steady internet connection is required.
- The system will simply display a string representing the emotion and not generate any response to address that emotion.
- The output will also contain graphs which will indicate the how the emotional state of a person has changed over time.

4.3 Major Constraints

- Any constraints that will impact the manner in which the software is to be specified, designed, implemented or tested are noted here.
- The sensors on the NAO robot must be fully operational
- Necessary python environment should be set up (preferably Anaconda)
- Necessary python packages installed (pip, numpy, sklearn)

4.4 Outcome

- Effective emotion recognition in the form periodic emotion labels displayed.
- Emotion recognition accuracy maintained by dynamically considering different inputs when one mode becomes unavailable.

4.5 Applications

- Development of an affect-aware city
- Add security layer at public venues to detect malicious intent and deal with hostage situation effectively

- Measure response and ratings in focus groups (consumer response to commercials etc)
- Wearables that help autistics discern emotion etc.

4.6 Software Resources Required

1. Operating System: Linux 64-bit
2. IDE: Anaconda (Conda), Microsoft Visual Studio Code
3. Programming Language: Python 3.6
4. Libraries: numpy, sklearn, OpenCV, dlib, pyAudio, google-api-python-client
5. Framework for UI: Flask, Bootstrap HTML Framework, any web browser
6. Google Cloud API: account, key

4.7 Hardware Resources Required

4.8 Project Resources

- Required hardware resources for training classifiers are available.
- Licensed softwares and libraries are available.
- Datasets and libraries used are provided under licenses which allow free use for non-commercial purposes.
- The training of deep neural networks largely constitutes the time requirements for the projects involving deep learning.
- Datasets and libraries are provided with licenses which allow use for non-commercial purposes.

Sr.No.	Parameter	Minimum Requirement	Justification
1	CPU Speed	3 GHz	Ensures that modules are trained in a timely manner
2	GPU	4 GB+RAM	Ensures that modules are trained in a timely manner
3	RAM	8GB	To process large amount of training data
4	NAO Robot Camera+Mic	320x240, 16000Hz	Getting accurate input

Table 4.1: Hardware Requirments

- GNU GPL license, MIT license and Apache license allow use of datasets and libraries for non-commercial purposes. GNU GPL allows free use and distribution of software under its license, as long as it's derivatives follow the same licensing model.

4.9 Risk Management w.r.t. NP Hard analysis

4.9.1 Risk Identification

The possible risks are identified by considering the SRS, high-level and low-level design strategy. The risks are considered by evaluating the risk scenarios against various design, development and testing conditions. Please refer table 4.2 for all the risks.

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Inconsistent datasets	Low	High	High	High
2	Skewed training data	Low	Low	Low	Low
3	All three inputs noisy at once	Medium	Medium	Very High	High
4	Sarcasm in the interaction	Low	Medium	Medium	High
5	NAO interface restrictions	Low	Low	High	High

Table 4.2: Risk Table

Probability	Value	Description
High	Probability of occurrence is	$> 75\%$
Medium	Probability of occurrence is	$26 - 75\%$
Low	Probability of occurrence is	$< 25\%$

Table 4.3: Risk Probability definitions

Impact	Value	Description
Very high	$> 10\%$	Schedule impact or Unacceptable quality
High	$5 - 10\%$	Schedule impact or Some parts of the project have low quality
Medium	$< 5\%$	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

Table 4.4: Risk Impact definitions

4.9.2 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk.

Risk ID	1
Risk Description	Inconsistent Datasets
Category	Development Environment.
Source	Improper description and samples on the website
Probability	Low
Impact	High
Response	Data cleaning by validating audio/video files against file name
Strategy	Data preprocessing
Risk Status	Occurred and mitigated

Risk ID	2
Risk Description	Skewed Training Data
Category	Development Environment
Source	Diversity in dataset absent
Probability	Low
Impact	High
Response	Dropping the dataset in favour of a more comprehensive one
Strategy	Validating as much of the dataset as possible
Risk Status	Identified

Risk ID	3
Risk Description	All inputs noisy at once
Category	Development Environment
Source	Software Design Specification documentation review.
Probability	Medium
Impact	High
Response	Skip the accepted inputs until one input becomes accurate to ensure real-time response
Strategy	Dynamically switching between inputs and displaying previously identified emotion
Risk Status	Identified and Mitigated

Risk ID	4
Risk Description	Sarcasm present in the interaction
Category	Development and Testing
Source	This was identified early during development.
Probability	Low
Impact	High
Response	
Strategy	Disable unnecessary sensors, use NAO only as a front-end
Risk Status	Identified and Mitigated

Risk ID	5
Risk Description	Processes starve due NAO's internal computations
Category	Technology
Source	This was identified late during development.
Probability	High
Impact	Very High
Response	Analyzing even the subtle differences in the input from modules
Strategy	Identifying sudden emotion changes/ high disparity between outputs from two modules
Risk Status	Identified

Chapter 5

Software requirement specification

5.1 Introduction

5.1.1 Purpose and Scope of Document

The purpose of the Software Requirements Specification document is to build a system to effectively recognize emotions at real-time to develop an emotionally aware robot as well as provide help to autistic people. The approach used for this task is a novel way of considering three input modules instead of just one. The different inputs used are the video frames, audio samples and the speech text. This document aims to identify the different and distinct design elements of the project as well as articulate the various requirements needed to make this project successful. The scope is limited to recognizing the emotions and not responding or generating any context related response. Also, the emotions considered for recognizing are limited to a pre-defined mutually-exclusive set of Happiness, Anger, Neutral, Sadness. The UML standard is adopted to represent the various diagrams which include Use-Case Diagram, Activity Diagram, Class Diagram, Sequence Diagram, State Diagram.

5.1.2 Intended Audience and Reading Suggestion

This project is a prototype for an emotion recognition system and it is restricted within the college premises. The project has been implemented for research purposes under the guidance of the Computer Department H.O.D Dr. V.Y. Kulkarni. This project is useful for individuals looking for a way to perform emotion recognition to use it a targeted application as well as a way to process multimodal inputs simultaneously.

5.2 Usage Scenario

This section provides various usage scenarios for the system to be developed.

5.2.1 Use-cases

All possible use-cases for the system are presented in the following table:

Sr No.	Use Case	Description	Actors	Assumptions
1	Take audio,video samples	Audio is captured whenever there is a 5 sec continuous utterance, Video is captured when a face is detected	Camera, Mic(Modelling actors)	Camera, Mic is initialized
2	Emotion Recog using Video Module	Emotion is recognized by analyzing the orientation of the facial landmarks	Server	Face is detected
3	Emotion Recog using Audio Module	Enabled when energy of sample is above a threshold, analyzed using energy & MFCCs	Server	Audio energy above threshold
4	Emotion Recog using Text Module	Speech-to-text conversion is done, feeding transcript as feature vector to classifier	Server	The transcript does not contain out-of-place words.
5	Display emotion label	Final recognized emotion is displayed on the web interface	Web interface, Server	Appropriate weights are assigned previously
6	Perform Speech-To-Text	Converts audio sample into transcripts	Google API, Server	Internet connection available

Table 5.1: Use Cases

5.2.2 Use Case View

The following gives the Use Case Diagram for the proposed system.

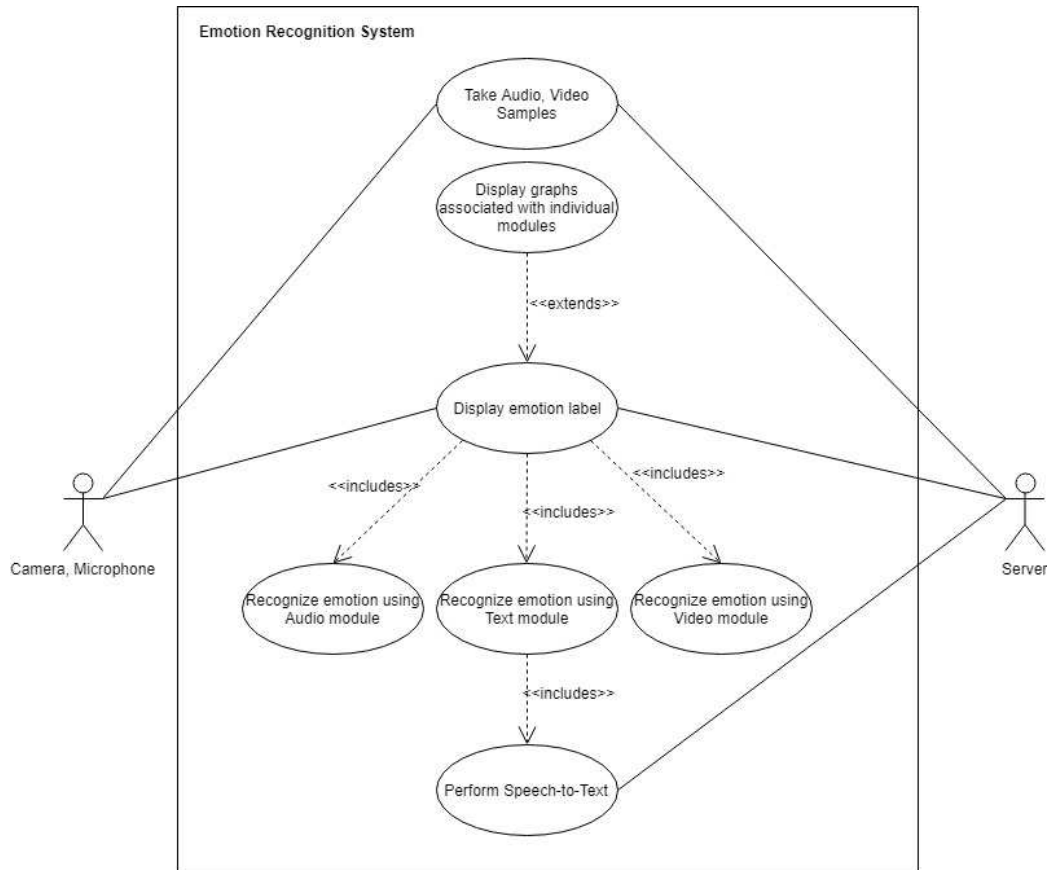


Figure 5.1: Use case diagram

5.3 Functional Model and Description

Inputs in the form of audio samples and video frames from a human through a camera and microphone. The System works on 3 distinct classifiers, each trained to handle a particular form of input. The audio input is given to the "Audio Recorder" whereas the video frames are given to the Video module. The Audio Recorder further sends the input to the Tone Analysis module and the Text Module. Each module runs their respective algorithm to recognize the emotions and assign their emotion label. The three labels are then sent to the web interface and the combined output is sent to the main process.

5.3.1 Data Flow Diagram

Data is in 3 distinct formats within the system which are video, audio and text. This data is captured by the input devices in the form of camera and microphone and sent to the server as .wav file for audio, .mp4 for video and transcribed text from the .wav audio file. The processing is carried out on this input and data is sent to the web interface. The data is also stored in an intermediate storage to enable access for the main process as well as the individual modules. Refer to figure 5.2 for the Dataflow Diagram of the proposed system using UML notations.

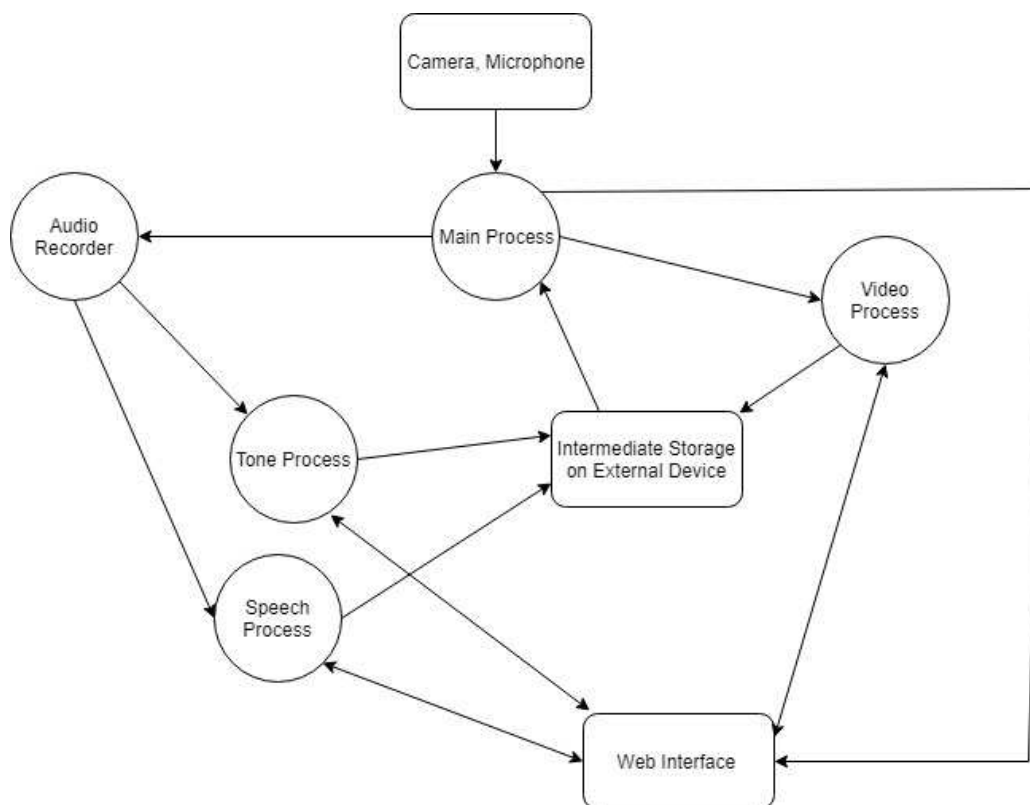


Figure 5.2: Dataflow diagram

5.3.2 Class Diagram

There are six classes considered in the system.

- **FlaskApp**: This class is responsible for starting the web server, reading data from pipes and serving HTTP requests to render the web client in a browser.
- **main** : This class is the core backend class of our system. It is responsible for spawning the 4 main sub processes, writing emotion probabilities to pipes and calculating the weighted sum for overall emotion.
- **video** : This class is responsible for reading images(frames) from input camera or input file, detecting faces in the frame and finally assigning an emotion to the face.
- **audioRecorder** : This class is responsible for getting an audio stream from microphone or input file and passing that on to tone and speech classes separately.
- **tone** : This class is responsible for extracting features from the input sound stream and assigning an emotion based on given features.
- **speech** : This class is responsible for converting given input stream to words using Google Speech-to-Text API, and assigning an emotion based on given words.

Refer to figure 5.3 for the Class Diagram of the proposed system using UML notations.

5.3.3 Activity Diagram

The system starts with initializing input devices and capturing input. The input is then sent to a main process which the parent process for the entire system. The main process then spawns two processes for video and audio respectively. The video process is responsible for detecting video emotion using the video frames. The audio process further spawns two processes for audio emotion detection and text emotion detection. Each module detects an emotion and sends the results to the web interface. According to the weights assigned previously, the final emotion is identified and displayed. The system then can either perform the same steps continuously, or exit

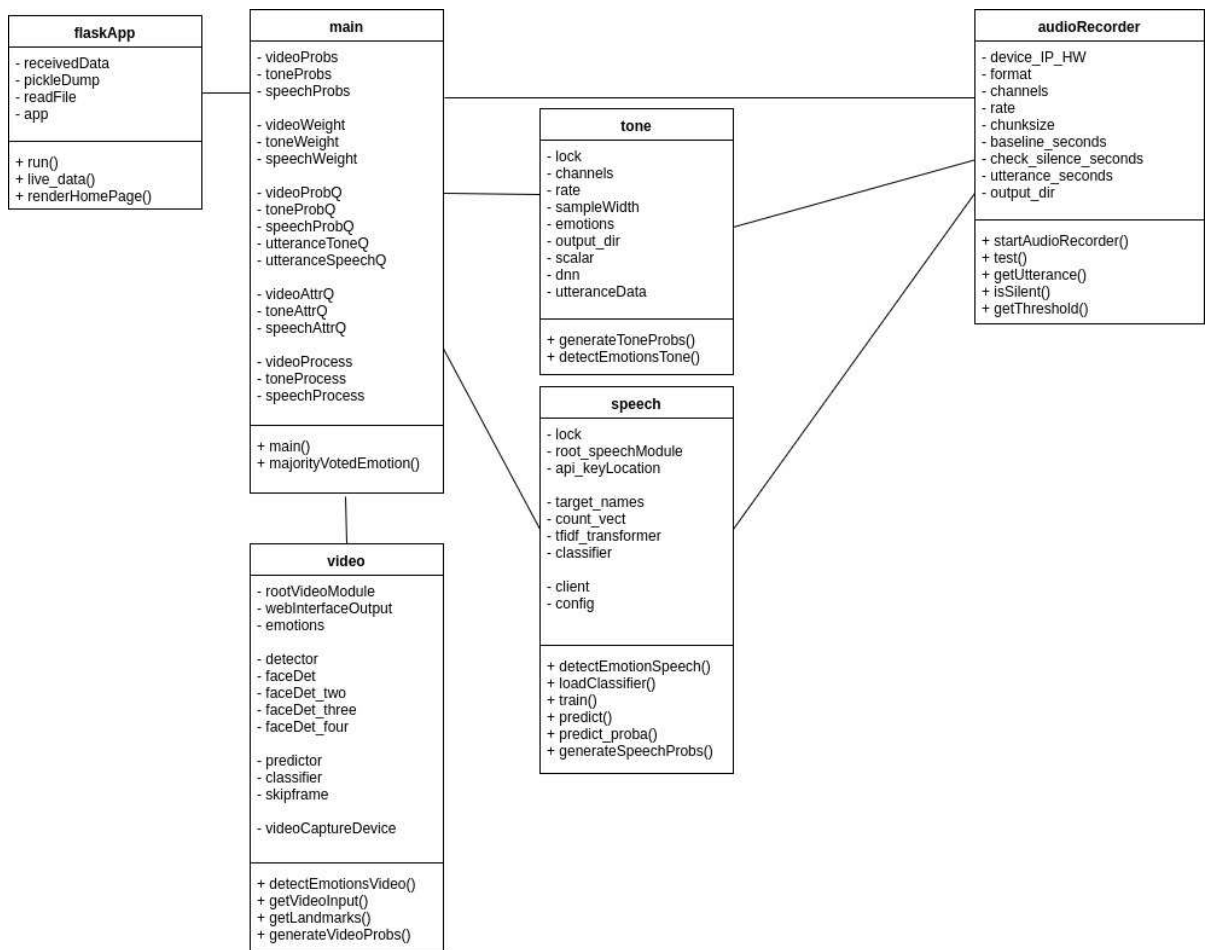


Figure 5.3: Class Diagram

depending on the user. Refer to figure 5.4 for the Activity Diagram of the proposed system using UML notations.

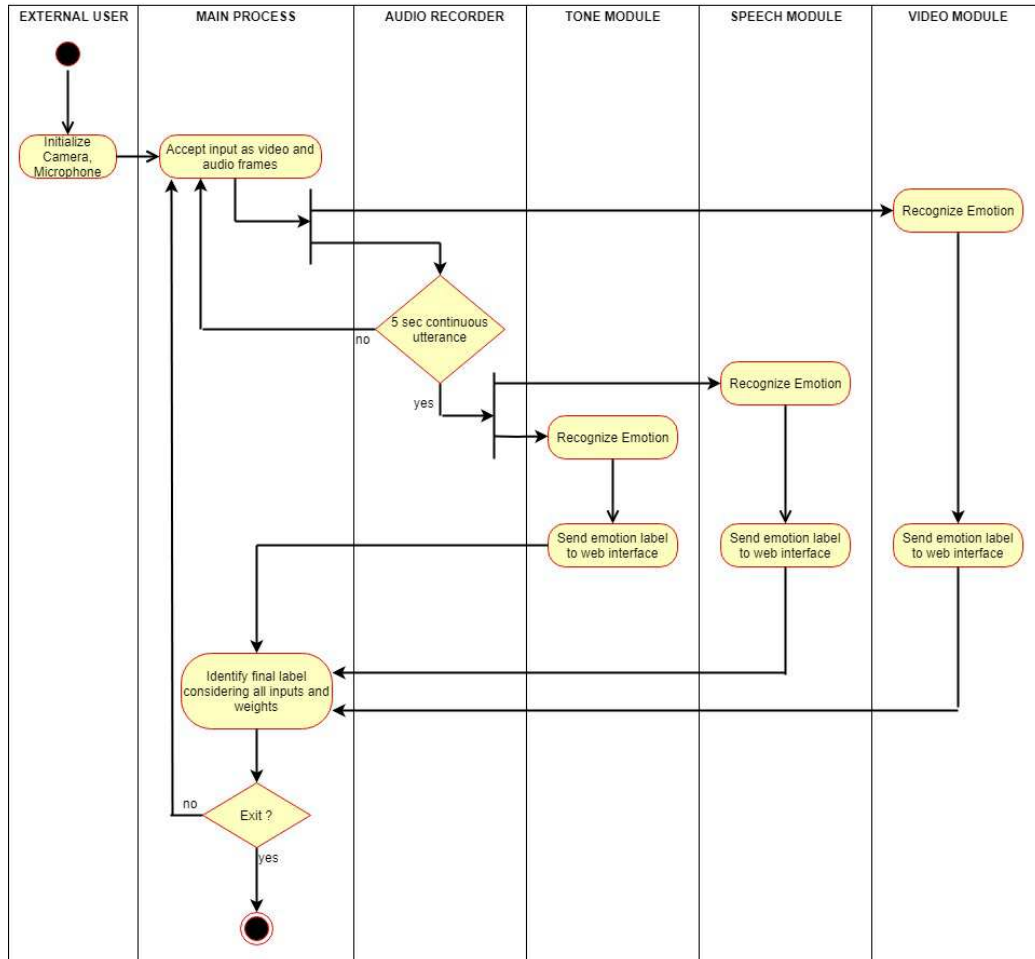


Figure 5.4: Activity Diagram

5.3.4 State Diagram

The system works in a user controlled infinite loop and tries to recognize emotions continuously. It goes through various states in a loop from getting input, spawning internal processes, detecting emotions on individual modules, collecting results and displaying necessary data until the system is

stopped by the user. Refer to figure 5.5 for the State Diagram of the proposed system using UML notations.

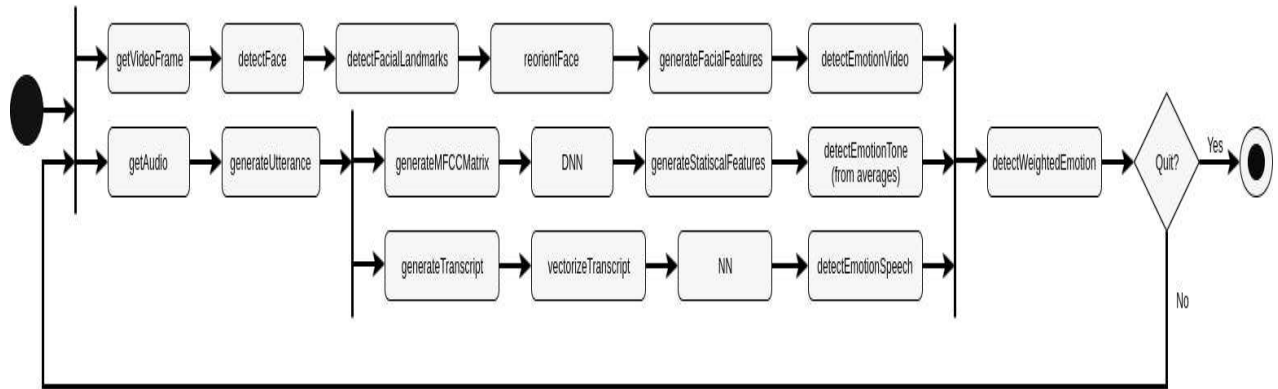


Figure 5.5: State Diagram

5.3.5 Deployment Diagram

The system contains 3 components which are the input device, the external server to carry out the processing and the client which can view the results. The input device captures the input and sends it to the server via a USB interface. The server does the processing and sends the final emotion label as well as the individual data of different modules to the client using http interface and default port 5000. The client uses a web browser to view the emotion label as well as the real-time graphs. Refer to figure 5.6 for the Deployment Diagram of the proposed system using UML notations.

5.3.6 Non Functional Requirements

- The system should be near real-time.
- System should be able to handle multiple computations executing simultaneously and potentially interacting with each other.

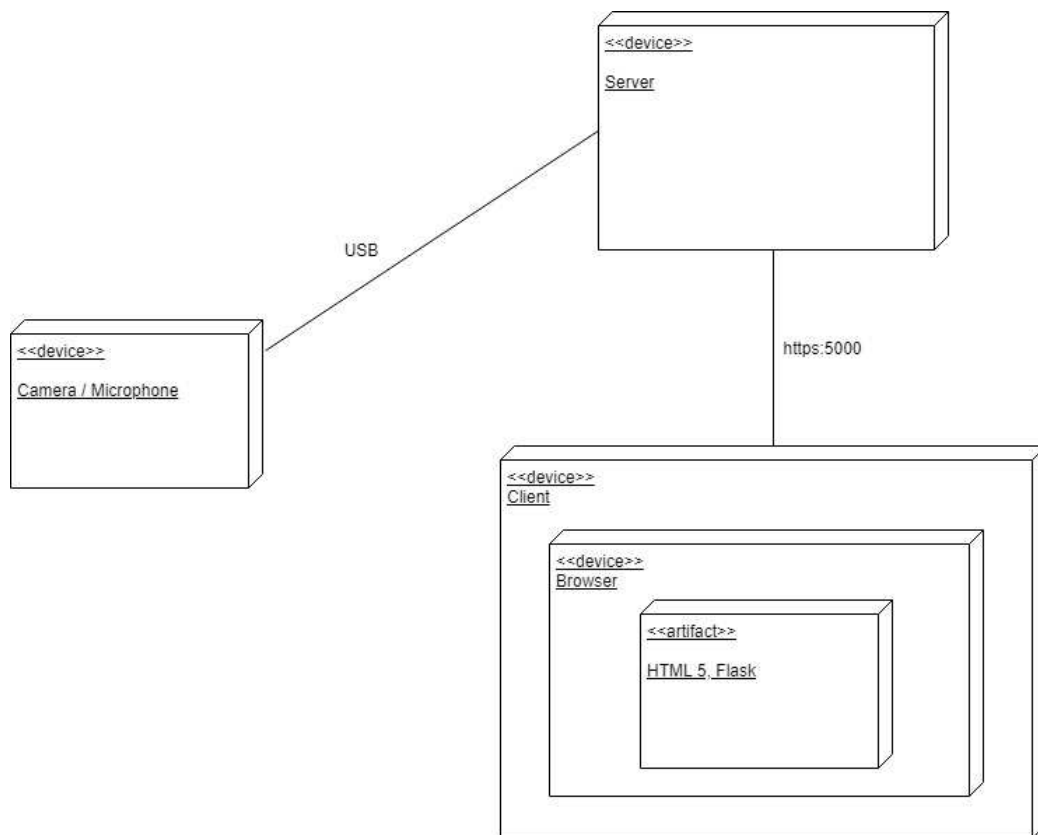


Figure 5.6: Deployment Diagram

- System should be able to store data to be communicated between processors.
- The file system should have enough space for interprocessor communication.
- The system will require continuous and frequent updates for the training models to improve accuracy.
- Proper and insightful documentation is to be maintained.

5.4 Architectural Design

The project involves two main subsystems - the processing server, and the web interface. The processing server takes the input from video or inputs from camera and mic, analyses the incoming audio and video. It is comprised of 3 components, the video module for analysing facial features, tone module and speech module for audio analysis. Each of the component returns its output to the parent process. The main process along with the subprocesses for these 3 modules communicate with the web interface. Web interface displays the visual reports for the data generated from the processing server in real-time. Refer to figure 5.7

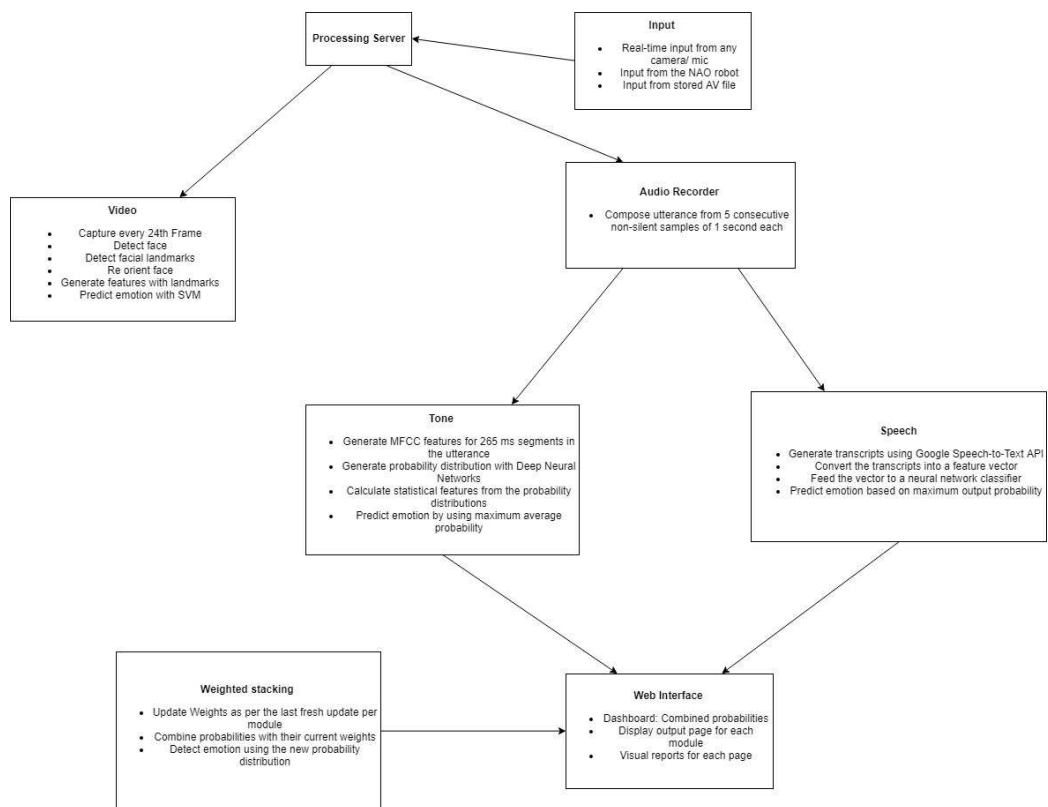


Figure 5.7: Architecture diagram

Chapter 6

Project Implementation

6.1 Methodologies/Algorithm Details

6.1.1 Algorithm/Pseudo Code

Following are the algorithms/pseudo codes used for the different modules which make up the system.

- **Main Process**

The main process is responsible for spawning all sub processes, updating each module's weight and sending detected emotion values to the web server.

First, we initialize the 3 modules weights with default = 1. Then we start each sub process:

VIDEO: Takes an image frame as input and outputs detected emotion. Detailed algorithm is provided later in this document.

AUDIORECORDER: Initializes and starts recording sound from the mic. Used by both TONE and SPEECH sub processes.

TONE: Takes a sound signal as input, detects human speech and outputs detected emotion. Detailed algorithm is provided later in this document.

SPEECH: Takes a sound signal as input, detects spoken words and outputs detected emotion. Detailed algorithm is provided later in this document.

For every iteration of the loop, each module will either detect an emotion or it will not. If an emotion is detected, the module's weight will be set to 1 and newly detected emotion probabilities will be used to calculate overall emotion. If any module does not detect any emotion (example: no face in frame or only noise being recorded),

that module's weight will be decreased according to a linear function which takes into account module specific features that might influence the module's accuracy.

Finally, by using a weighted sum a combined emotion score is calculated.

The algorithm is as follows:

```
initialize videoWeight = 1
initialize toneWeight = 1
initialize speechWeight = 1

initialize and start video subprocess
initialize and start audioRecorder subprocess
initialize and start tone subprocess
initialize and start speech subprocess
```

Repeat till exit:

```
If new videoProbabilities are predicted:
sendToGraph(videoProbs)
videoWeight = 1
else:
decrease videoWeight by linear function
```

```
If new toneProbabilities are predicted:
sendToGraph(toneProbs)
toneWeight = 1
else:
decrease toneWeight by linear function
```

```
If new speechProbabilities are predicted:
sendToGraph(speechProbs)
speechWeight = 1
else:
decrease speechWeight by linear function
```

- **Video Module:** The video module sub process is responsible for detecting emotion by looking at facial features in an image. Thus, it

receives an image frame as input and outputs an emotion if a face is detected.

To detect faces in a frame, we use OpenCV's Haar cascade classifiers. We have provided more than one classifier to be able to detect faces at slight angles to the camera.

After a face is detected, we extract features from it to feed to the emotion classifier. To do this, we detect 68 track key points on the face. Refer to figure 6.1 for distribution of the 68 landmarks on the face and a corresponding example in figure 6.2

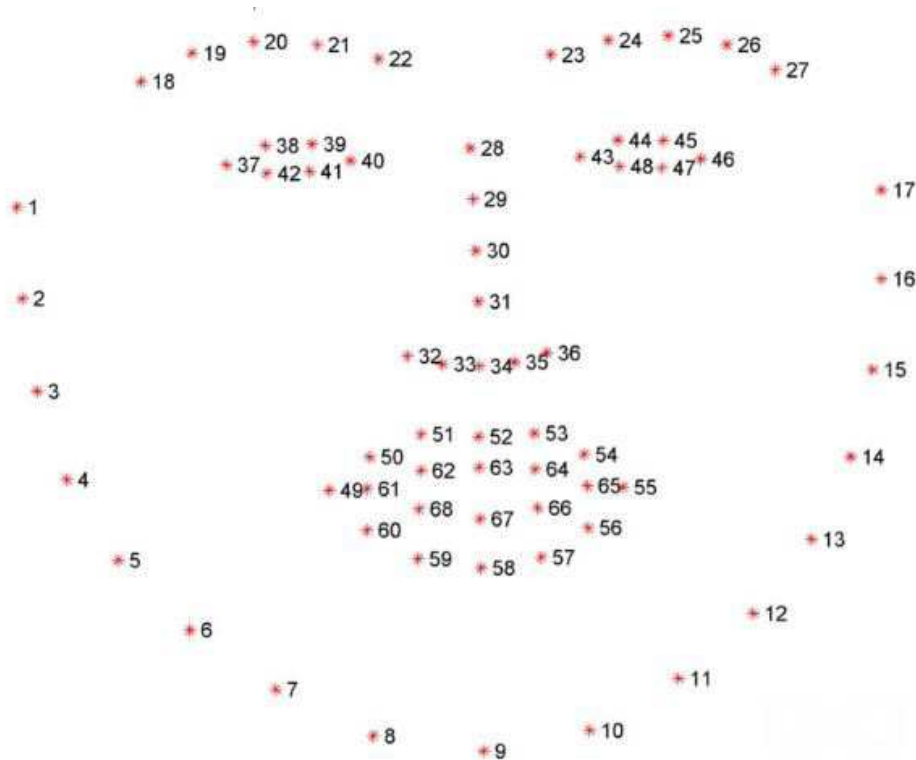


Figure 6.1: Facial Landmarks for detection

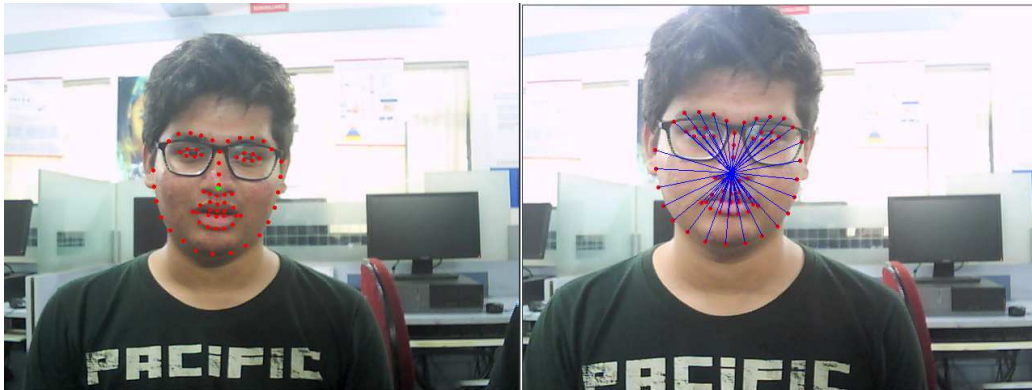


Figure 6.2: Facial Landmarks for detection Example

These points delineate various facial features like eyebrows, nose etc. The mean (X,Y) co-ordinates of these points is calculated. Finally, a feature vector is created which consists of:

1. Distance of each point from the mean.
2. Angle made by each point relative to the mean.

This feature vector is then fed to a classifier which calculates probabilities for each emotions based on the given features. The classifier has already been trained by us using the extended COHN-KANADE dataset.

Now to increase performance, we perform this process on every n th frame. We can afford to skip frames without losing accuracy because facial expressions do not change instantly over consecutive frames - they change gradually over several frames. By default, we set the skipframe value as 24.

- **Audio Module:**

```
initialize the variables RATE
load classifier parameters for segment emotion detection
load normalizer parameters, if needed by the classifier
utteranceCount = 0
repeat till exit:
  retrieve utterance
  frameFeatureMatrix = extractFeatures(utterance)
```

```

topSegmentIndices = getTopEnergySegmentIndices(utterance, RATE)
topSegmentFeatureMatrix =
getSegmentFeaturesUsingIndices(frameFeatureMatrix,
topSegmentIndices)
topSegmentFeatureMatrix = normalize(topSegmentFeatureMatrix)
segmentProbabilities = predict(topSegmentFeatureMatrix)
avgSegmentProbabilities = mean(segmentProbabilities)
emotion = max(segmentProbabilities)
communicate avgSegmentProbabilities, emotion to main process
utteranceCount += 1

```

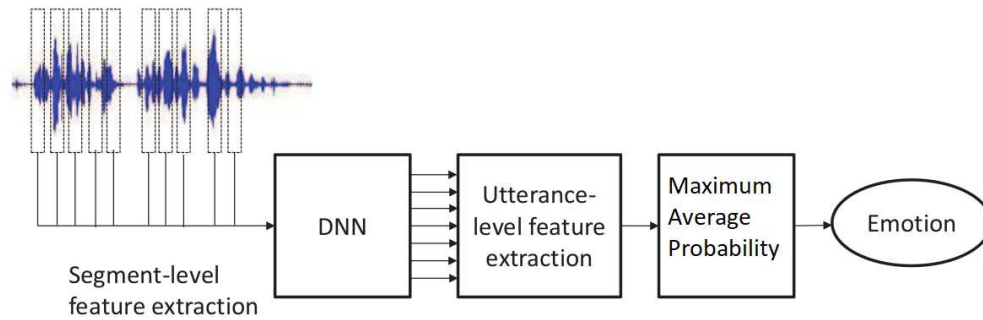


Figure 6.3

- **Speech Module:**

Feature Selection: First we extract the entire vocabulary V from our dataset D . For a given emotion class c , we compute a utility measure $A(t, c)$ for each term t in vocabulary V . We then select k terms having the highest values of $A(t, c)$ as our features. The values of these features in a feature vector will be the normalized frequencies of the corresponding words in a document.

6.1.1.0.1 We use mutual information (MI) as our utility measure: $A(t, c) = I(U_t; C_c)$. MI measures how much the presence/absence of a term contributes to making the correct classification decision.

Formally:

$$I(U; C) = \sum_{e_t \in (1,0)} \sum_{e_c \in (1,0)} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)} \quad (6.1)$$

where U is a random variable that takes values $e_t = 1$ (the document contains the term t) and $e_t = 0$ (the document does not contain t), C is a random variable that takes values $e_c = 1$ (the document is in class c) and $e_c = 0$ (the document is not in class c).

The pseudo code for the speech analysis module is as follows:

```
load trained text classifier
start transcriber
transcript_record =
window_time = k
Repeat till exit:
Read audio from audiorecorder
Convert audio to transcript using transcriber
Add transcript to transcript_record
Generate TF-IDF normalized feature vector from transcript_record
Detect emotion label probabilities of generated...
...feature vector using trained text classifier
Remove data from transcript_record older than window_time
```

6.2 Reports for classifier training process

- **Tone Analysis:**

- **DataSet used:**

The Interactive Emotional Dyadic Motion Capture (IEMOCAP) database is an acted, multimodal and multispeaker database. It contains approximately 12 hours of audiovisual data, including video, speech, motion capture of face, text transcriptions. It consists of dyadic sessions where actors perform improvisations or scripted scenarios, specifically selected to elicit emotional expressions. IEMOCAP database is annotated by multiple annotators into categorical labels, such as anger, happiness, sadness, neutrality, as well as dimensional labels such as valence, activation and dominance. The detailed motion capture information, the interactive setting to elicit authentic emotions, and the size of the database make this corpus a valuable addition to the existing databases in the community for the study and modeling of multimodal and expressive human communication.

- **TONE UTTERANCES BEFORE COMBINING:** Refer to figure 6.4. For Tone analysis, we found that the dataset distribution was skewed.

- * Anger 1103
 - * Disgust 2
 - * Excitement 1041
 - * Fear 40
 - * Frustration 1849
 - * Happiness 595
 - * Neutral 1708
 - * Other 3
 - * Sad 1084
 - * Surprise 107
 - * total : 7532

- **TONE UTTERANCES AFTER COMBINING:** Refer to figure 6.5. To balance the dataset, we combined emotions with

similar annotations:

- * ang fru 2952
- * happ exc sur 1743
- * neutral 1708
- * sadness fear 1124

Dataset distribution for utterances - Before combining categories

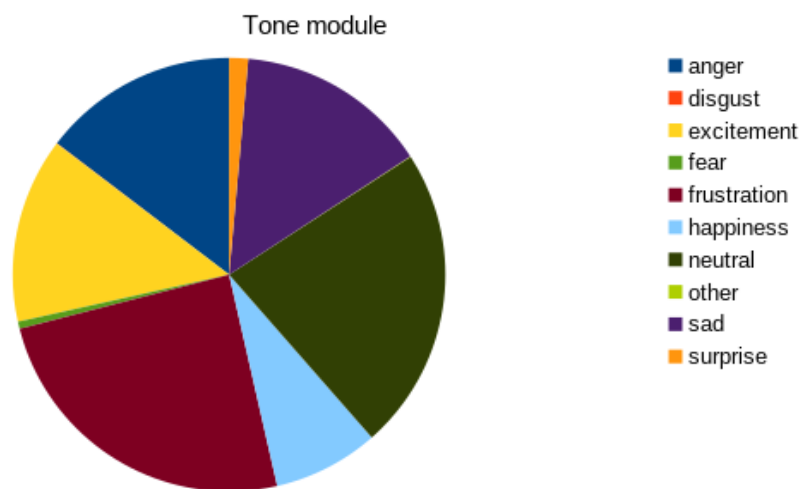


Figure 6.4: Tone Utterances before combining labels

- **Training and Validation report:** python trainDNN.py Dataset loaded into dataframe... X and y loaded.... Training and testing sets created... X_{train} and X_{test} normalized... Training DNN... Iteration 1, loss = 1.19501334
Validation score: 0.495714
Iteration 2, loss = 1.02921240
Validation score: 0.579298
Iteration 3, loss = 0.83500359
Validation score: 0.659660

Dataset distribution of utterances after combining categories

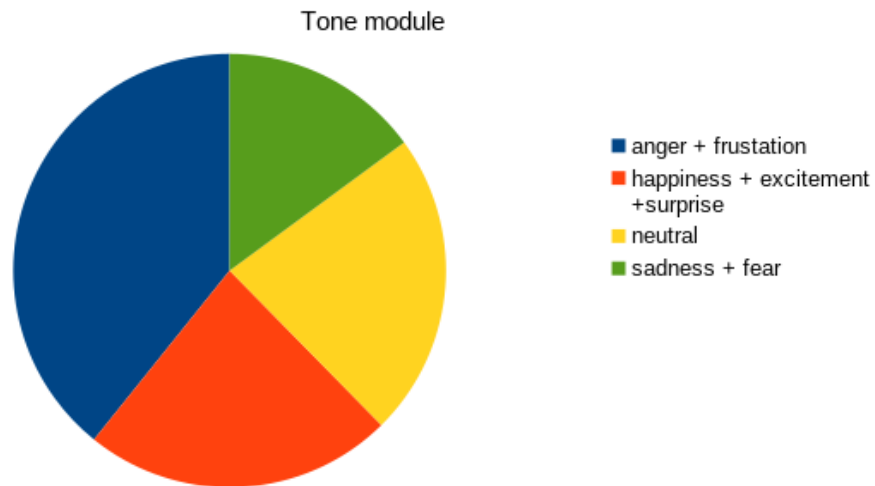


Figure 6.5: Tone Utterances after combining labels

Iteration 4, loss = 0.63884454
Validation score: 0.723208
Iteration 5, loss = 0.48286869
Validation score: 0.761533
Iteration 6, loss = 0.37261158
Validation score: 0.798439
Iteration 7, loss = 0.29365010
Validation score: 0.815581
Iteration 8, loss = 0.23836988
Validation score: 0.833706
Iteration 9, loss = 0.20316889
Validation score: 0.844625
Iteration 10, loss = 0.17408951
Validation score: 0.842660
Iteration 11, loss = 0.16278715
Validation score: 0.854015
Iteration 12, loss = 0.14406390
Validation score: 0.864825
Iteration 13, loss = 0.13786938
Validation score: 0.870394
Iteration 14, loss = 0.13181086
Validation score: 0.872304

Iteration 15, loss = 0.12003902
 Validation score: 0.875908
 Iteration 16, loss = 0.11500236
 Validation score: 0.874707
 Iteration 17, loss = 0.11445793
 Validation score: 0.878201
 Iteration 18, loss = 0.10277859
 Validation score: 0.881858
 Iteration 19, loss = 0.10925041
 Validation score: 0.883988
 Iteration 20, loss = 0.09472984
 Validation score: 0.888246
 Iteration 21, loss = 0.10075731
 Validation score: 0.887973
 Iteration 22, loss = 0.09333517
 Validation score: 0.884370
 Iteration 23, loss = 0.09320555
 Validation score: 0.886335
 Validation score did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
 DNN trained in 234.56247210502625 seconds ...
 TRAINING SET SCORE : 0.965337
 TESTING SET SCORE : 0.887380
 CONFUSION MATRIX for testing data :

```

[[10097   408   568   375]
 [  549 10147   369   382]
 [  447   201 10331   469]
 [  544   319   526 10059]]
  
```

CLASSIFICATION REPORT for testing data :

	precision	recall	f1-score	support
0	0.87	0.88	0.87	11448
1	0.92	0.89	0.90	11447
2	0.88	0.90	0.89	11448
3	0.89	0.88	0.88	11448
avg / total		0.89	0.89	0.89 45791

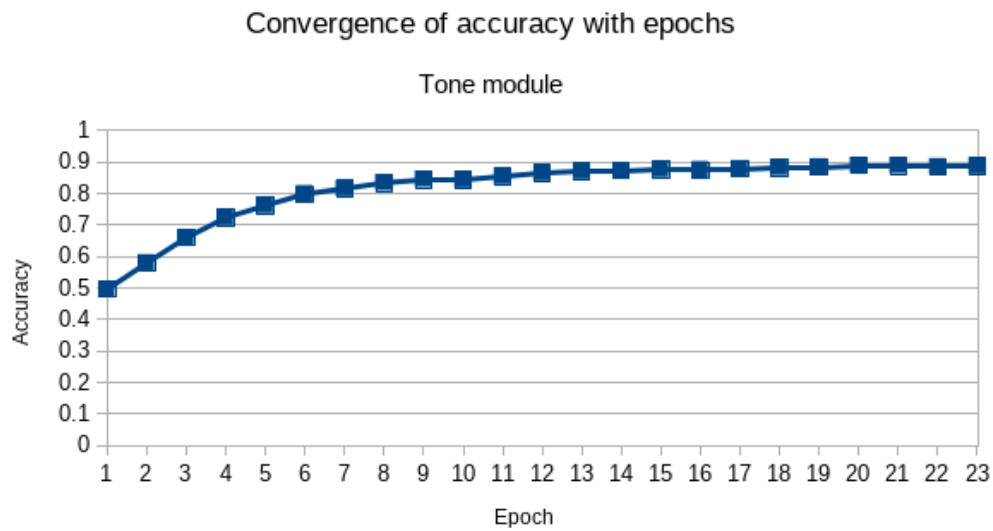


Figure 6.6: Learning Curve

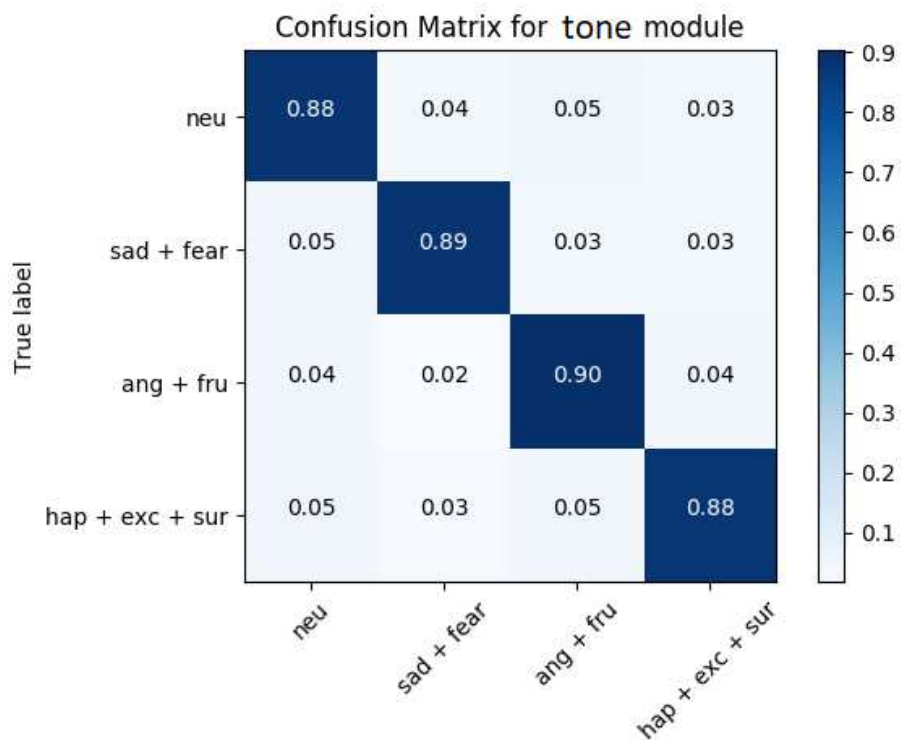


Figure 6.7: Confusion Matrix

- **Video Module:**

- **Dataset used:**

The Cohn-Kanade AU-Coded Facial Expression Database is for research in automatic facial image analysis and synthesis and for perceptual studies. Cohn-Kanade is available in two versions and a third is in preparation. We have used the second version of this dataset, referred to as CK+. It includes both posed and non-posed (spontaneous) expressions and additional types of metadata. The target expression for each sequence is fully FACS coded. In addition validated emotion labels are present in the metadata. Thus, sequences may be analyzed for both action units and prototypic emotions. Additionally, CK+ provides protocols and baseline results for facial feature tracking and action unit and emotion recognition.

- **VISUAL CORPUS DISTRIBUTION:** We trained on 6 emotions present in the dataset with distribution as follows:

Anger 45
Disgust 58
Happiness 69
Neutral 117
Sadness 28
Surprise 83

- **Training and Validation Report:**

Refer to figure 6.9 and 6.10

```
python \classifier_for_metrics.py\  
Making sets 0  
working on anger  
working on disgust  
working on happiness  
working on neutral  
working on sadness  
working on surprise  
training SVM linear 0  
getting accuracies 0
```

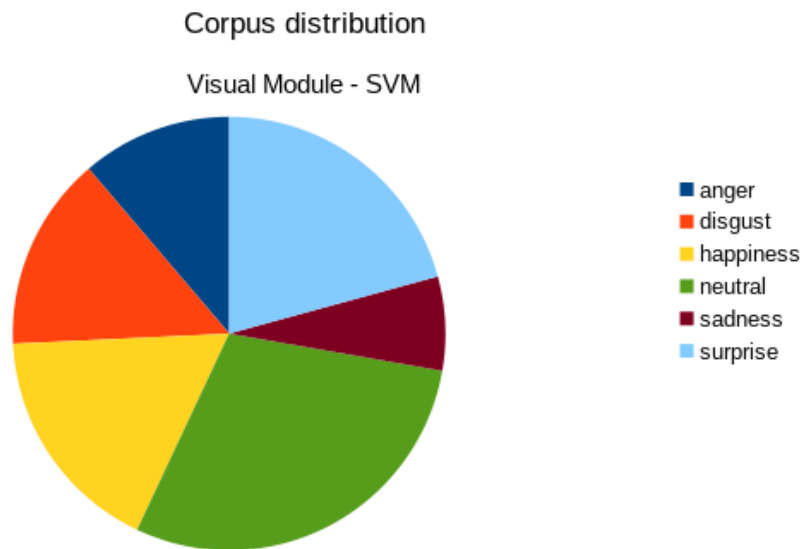


Figure 6.8: Visual Corpus Distribution

```
linear: 0.8181818181818182
Making sets 1
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 1
getting accuracies 1
linear: 0.8181818181818182
Making sets 2
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 2
getting accuracies 2
linear: 0.7922077922077922
Making sets 3
```


working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 3
getting accuracies 3
linear: 0.8701298701298701
Making sets 4
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 4
getting accuracies 4
linear: 0.8051948051948052
Making sets 5
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 5
getting accuracies 5
linear: 0.8961038961038961
Making sets 6
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 6
getting accuracies 6
linear: 0.7662337662337663
Making sets 7
working on anger

```

working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 7
getting accuracies 7
linear: 0.8701298701298701
Making sets 8
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 8
getting accuracies 8
linear: 0.8441558441558441
Making sets 9
working on anger
working on disgust
working on happiness
working on neutral
working on sadness
working on surprise
training SVM linear 9
getting accuracies 9
linear: 0.7792207792207793
[0.8181818181818182, 0.8181818181818182, 0.7922077922077922,
0.8701298701298701, 0.8051948051948052, 0.8961038961038961,
0.7662337662337663, 0.8701298701298701, 0.8441558441558441,
0.7792207792207793]
Mean value lin svm: 0.825974025974026
[[ 6  2  0  1  0  0]
 [ 1 10  0  0  0  0]
 [ 0  0 13  0  0  0]
 [ 2  0  0 16  4  1]
 [ 1  0  0  1  3  0]
 [ 0  0  1  1  2 12]]

```

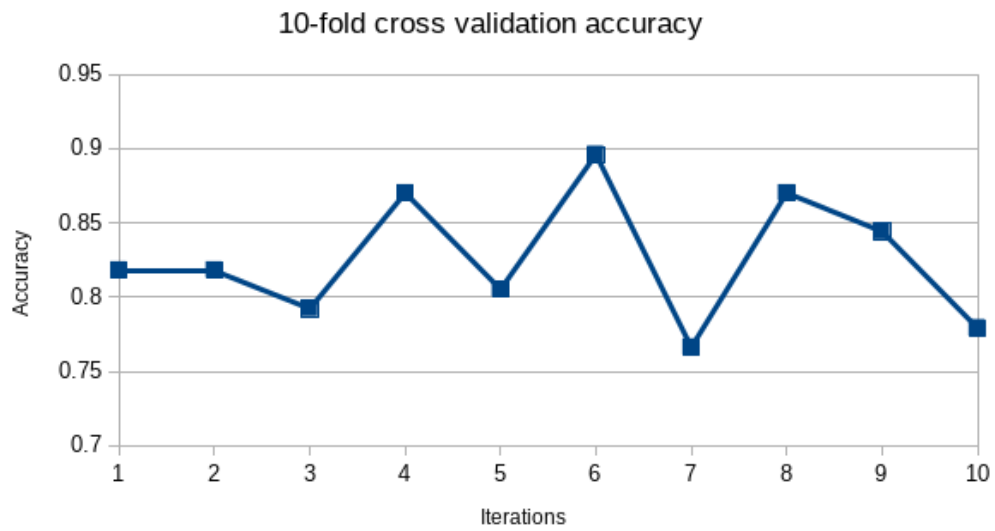


Figure 6.9: Learning Curve

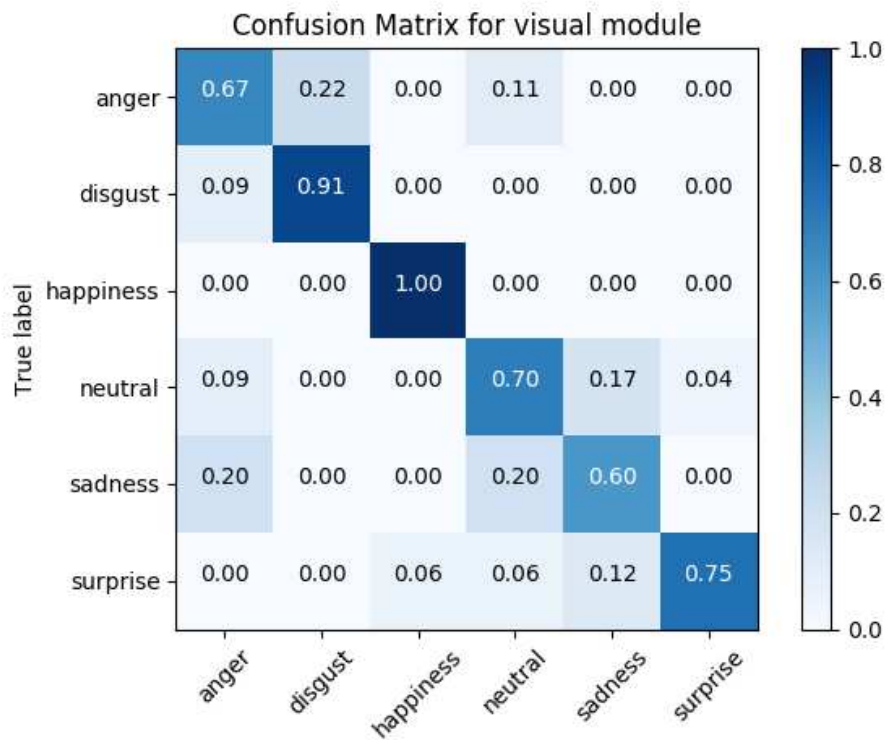


Figure 6.10: Confusion Matrix

- **Text Analysis:**

- **DataSet used:**

The Interactive Emotional Dyadic Motion Capture (IEMOCAP) database is an acted, multimodal and multispeaker database. It contains approximately 12 hours of audiovisual data, including video, speech, motion capture of face, text transcriptions. It consists of dyadic sessions where actors perform improvisations or scripted scenarios, specifically selected to elicit emotional expressions. IEMOCAP database is annotated by multiple annotators into categorical labels, such as anger, happiness, sadness, neutrality, as well as dimensional labels such as valence, activation and dominance. The detailed motion capture information, the interactive setting to elicit authentic emotions, and the size of the database make this corpus a valuable addition to the existing databases in the community for the study and modeling of multimodal and expressive human communication. For speech analysis, we use the transcripts of utterances having the same label by atleast 2 human annotaters.

- **Test Metrics:**

No. of utterances combined	Avg. No. of words	Accuracy	Precision	F1-Score
1	12.0645	0.58125	0.58	0.58
2	23.1291	0.73	0.72	0.72
3	34.1937	0.74	0.74	0.73
4	45.2583	0.82	0.77	0.79
5	56.3229	1	0.79	0.88
6	67.3875	0.89	0.89	0.89

Table 6.1: Risk Impact definitions

As the size of the input changes, following is observed with

respect to accuracy, precision and recall:

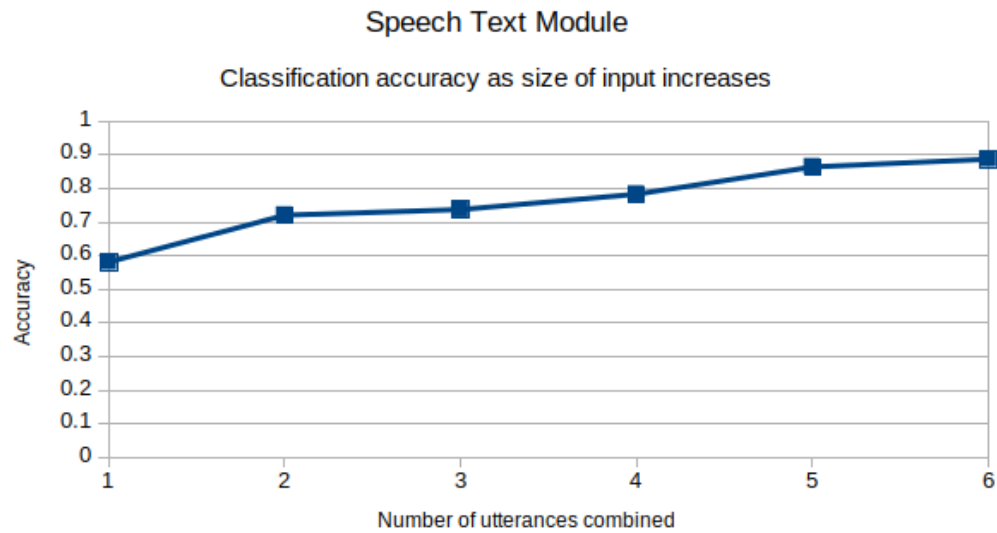


Figure 6.11: Accuracy with change in input size

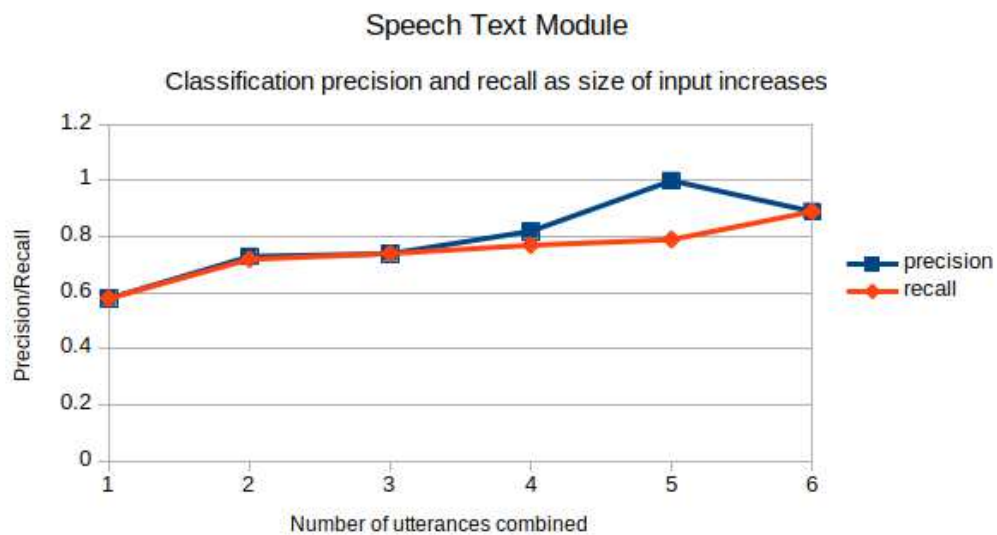


Figure 6.12: Precision and Recall with change in input size

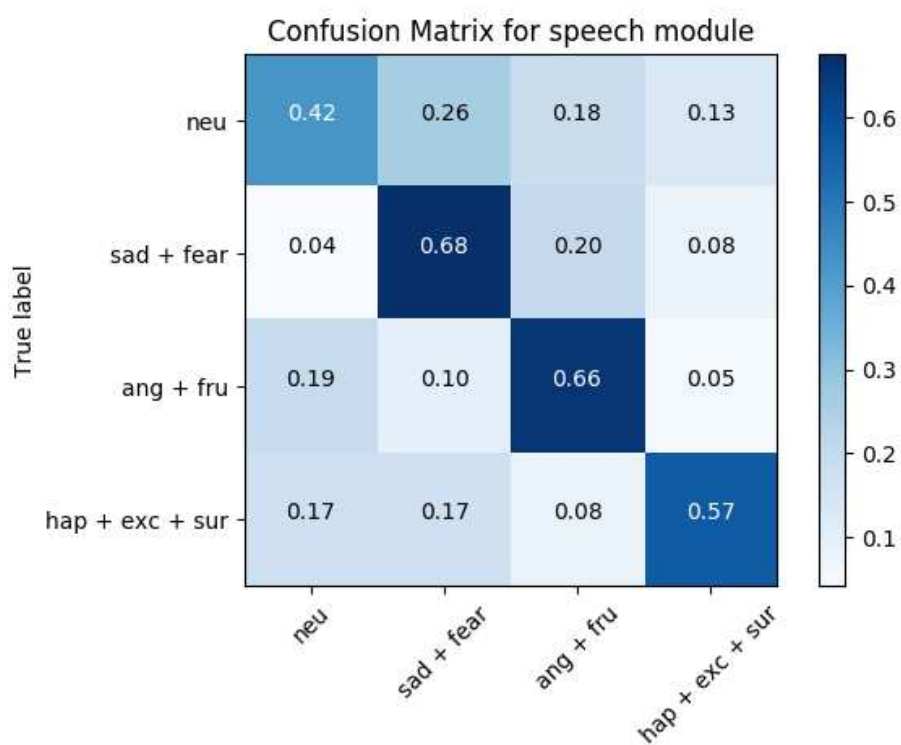


Figure 6.13: Speech Module Confusion Matrix (cm for 12 average word count)

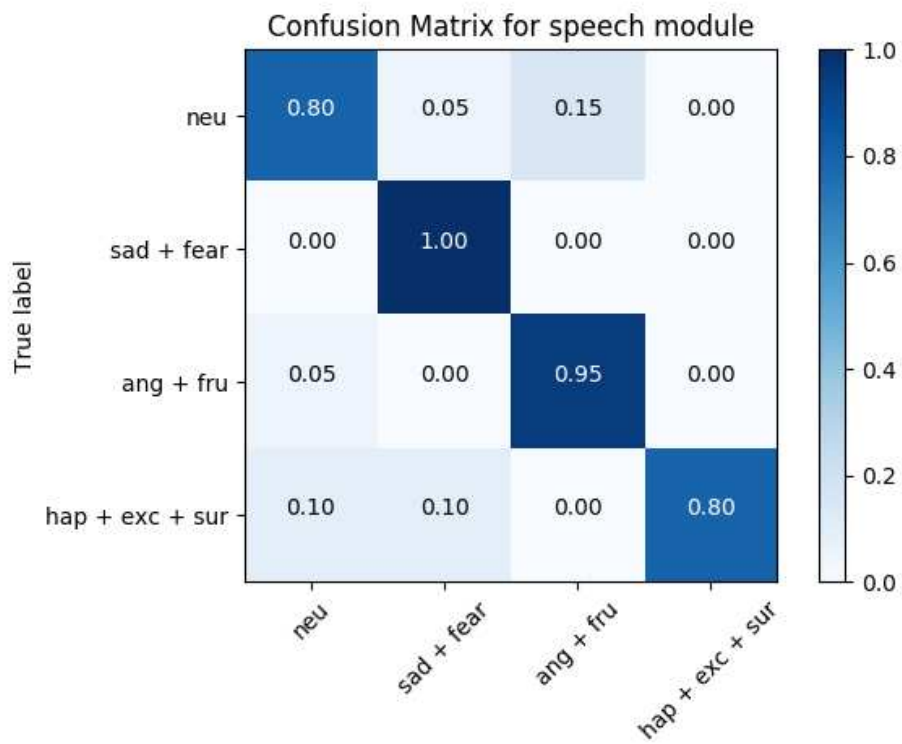


Figure 6.14: Speech Module Confusion Matrix (cm for 67 average word count)

Chapter 7

Software Testing

7.1 Type of Testing Used

- **Unit Testing:**

- **Introduction:** Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. It is a type of White Box Testing. Unit Testing is the first level of software testing and is performed prior to Integration Testing. It is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.
- **Unit Testing Approach:** We have performed Unit testing separately on all modules by giving positive and negative inputs. Following is the approach used for unit test performed on the video module:

```
import unittest
from imagePredictor_test import predict_emotion

class Test(unittest.TestCase):

    def setUp(self):
        pass
```



```

def test_happiness(self):
    self.assertEqual(predict_emotion("unitTesting/
happiness.jpg"), "happiness")

def test_disgust(self):
    self.assertEqual(predict_emotion("unitTesting/
disgust.jpg"), "disgust")

def test_anger(self):
    self.assertEqual(predict_emotion("unitTesting/
anger.jpg"), "anger")

def test_neutral(self):
    self.assertEqual(predict_emotion("unitTesting/
neutral.jpg"), "neutral")

def test_surprise(self):
    self.assertEqual(predict_emotion("unitTesting/
surprise.jpg"), "surprise")

def test_sadness(self):
    self.assertEqual(predict_emotion("unitTesting/
sadness.jpg"), "sadness")

def test_failure(self):
    self.assertEqual(predict_emotion("unitTesting/
sadness.jpg"), "anger")

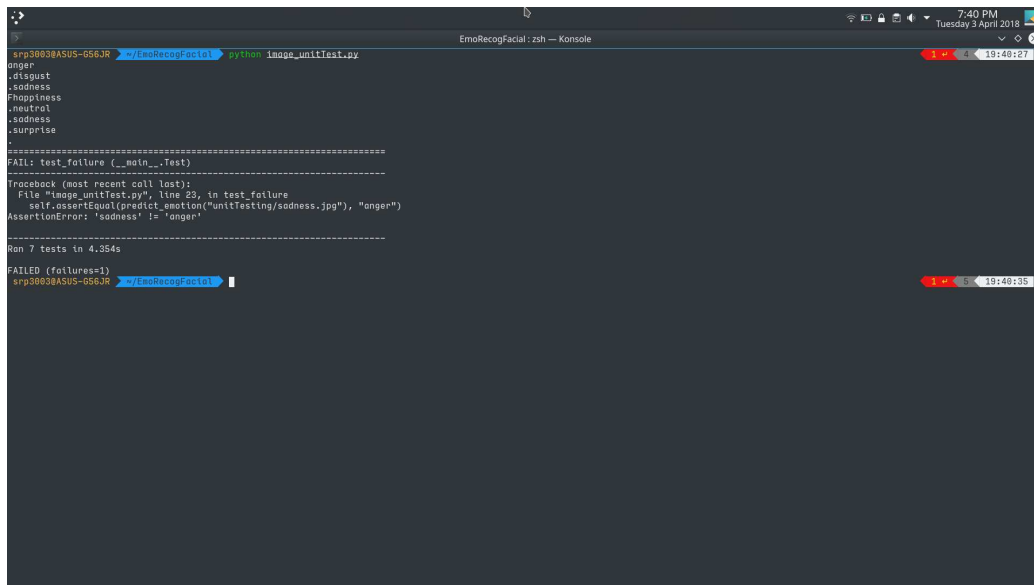
if __name__=="__main__":
    unittest.main()

```

For the output from the unit test, refer to figure 7.1

- **Integration Testing:**

- **Introduction:** Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates,



```
anger
.happiness
.sadness
.happiness
.neutral
.sadness
.surprise
=====
FAIL: test_failure (__main__._Test)
-----
Traceback (most recent call last):
  File "image_unitTest.py", line 23, in test_failure
    self.assertEqual(prediction("unitTesting/sadness.jpg"), "anger")
AssertionError: 'sadness' != 'anger'
=====
Ran 7 tests in 4.354s
FAILED (failures=1)
srp3083BASUS-G56JR > /EmoRecogFacial
```

Figure 7.1: Video Module Unit Test

applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

– **Project Structure:**

- * The project involves two main subsystems - the processing server, and the web interface.
- * The processing server takes the input from video or inputs from camera and mic, analyses the incoming audio and video. It is comprised of components, the video module for analysing facial features, tone module and speech module for audio analysis.
- * Each of the component returns its output to the parent process. The main process along with the subprocesses for these 3 modules communicate with the web interface.
- * Web interface displays the visual reports for the data generated from the processing server in real-time.

– **Integration Testing Approach:**

- * The project follows bidirectional integration testing.
- * The modules are tested as and when ready for integration.
- * This allows the flexibility of updates to the modules independently.

- * Stubs for each of the three modules have been created, which randomly generate data simulating the output of the modules. A driver main process in the server processes these module outputs and provides the data for web interface.
- * This allows for testing of the web interface even when modules are not ready for integration.

* **Driver:**

```

videoProcess = Process(target=generateVideoProbs,
args=(videoProbQ,))
toneProcess = Process(target=generateToneProbs,
args=(toneProbQ,))
speechProcess = Process(target=generateSpeechProbs,
args=(speechProbQ,))
videoProcess.start()
toneProcess.start()
speechProcess.start()
counter = 0
while(True):
    try:
        videoProbs = videoProbQ.get(block=False)
        videoProbUpdate = True
        videoWeight = 1.0
    except queue.Empty:
        videoProbUpdate = False
    if videoWeight >= 0.2:
        videoWeight -= 0.2
    try:
        toneProbs = toneProbQ.get(block=False)
        toneProbUpdate = True
        toneWeight = 1.0
    except queue.Empty:
        toneProbUpdate = False
    if toneWeight >= 0.2:
        toneWeight -= 0.2
    try:
        speechProbs = speechProbQ.get(block=False)
        speechProbUpdate = True
        speechWeight = 1.0
    except queue.Empty:
        speechProbUpdate = False

```

```

if speechWeight >= 0.2:
    speechWeight -= 0.2
    weights = [videoWeight, toneWeight, speechWeight]
    emotion, weightedAvgProbs = majorityVotedEmotion(videoProbs,
    toneProbs,
    speechProbs,
    weights)
    transmitArray = [weightedAvgProbs, weights, videoProbs,
    toneProbs, speechProbs, vid
    eoAttrs, toneAttrs, speechAttrs]
    pickle.dump(transmitArray, fp)
    counter += 1

```

*** Stub:**

- Video module stub:

```

def generateVideoProbs(videoProbQ):
    lock = Lock()
    randomGenerator = np.random.RandomState(seed=1)
    while(True):
        lock.acquire()
        try:
            videoProbs = randomGenerator.rand(6)
            videoProbQ.put(videoProbs, block=False)
        except queue.Full:
            pass
        finally:
            lock.release()
            time.sleep(3)

```

Tone module stub:

```

def generateToneProbs(toneProbQ):
    lock = Lock()
    randomGenerator = np.random.RandomState(seed=2)
    while(True):
        lock.acquire()
        try:
            toneProbs = randomGenerator.rand(6)
            toneProbQ.put(toneProbs, block=False)
        except queue.Full:

```

```
pass
finally:
    lock.release()
    time.sleep(6)
```

Speech module stub:

```
def generateSpeechProbs(speechProbQ):
    lock = Lock()
    randomGenerator = np.random.RandomState(seed=3)
    while(True):
        lock.acquire()
        try:
            speechProbs = randomGenerator.rand(4)
            speechProbQ.put(speechProbs, block=False)
        except queue.Full:
            pass
        finally:
            lock.release()
            time.sleep(5)
```

Chapter 8

Results

8.1 Outputs

8.1.1 Casting audition monologue by Breeze Woodson

- Source - <https://www.youtube.com/watch?v=ERfJB7j-BWc>
- Primary emotions are sadness and frustration in the monologue. The end of the video exhibits happiness when the actress introduces herself.
- Tested accuracies of the models :
 - video analysis accuracy : 53.19 %
 - tone analysis accuracy : 67.02 %
 - speech analysis accuracy : 0.0 %
 - majority voting accuracy : 71.28 %

8.1.2 Audition monologue "Getting out" by Marsha Norman

- Source - <https://www.youtube.com/watch?v=Zb9gzLU3hvQ>
- Primary emotions are anger, contempt and disgust.
- Tested accuracies of the models :
 - video analysis accuracy : 52.84 %

- tone analysis accuracy : 42.27 %
- speech analysis accuracy : 46.34 %
- majority voting accuracy : 72.35 %

8.1.3 Reactions - Unfair Mario gameplay

- Source - <https://www.youtube.com/watch?v=nBWimCDEo7o>
- Primary emotion are happiness, surprise, excitement, frustration.
- Tested accuracies of the models :
 - video analysis accuracy : 74.11 %
 - tone analysis accuracy : 29.41 %
 - speech analysis accuracy : 09.41 %
 - majority voting accuracy : 81.17 %

8.1.4 Macbeth monologue by Andrew Serkis

- Source - <https://www.youtube.com/watch?v=mkkWCmljMSA>
- Primary emotions are sadness, anger, frustration.
- Tested accuracies of the models :
 - video analysis accuracy : 79.20 %
 - tone analysis accuracy : 30.69 %
 - speech analysis accuracy : 49.50 %
 - majority voting accuracy : 79.20 %

8.1.5 Testing accuracy for real-time analysis

- Average video analysis accuracy - 64.84 %
- Average tone analysis accuracy - 42.35 %
- Average speech analysis accuracy - 26.31 %
- Average accuracy of stacked model - 76 %

8.2 Screenshots

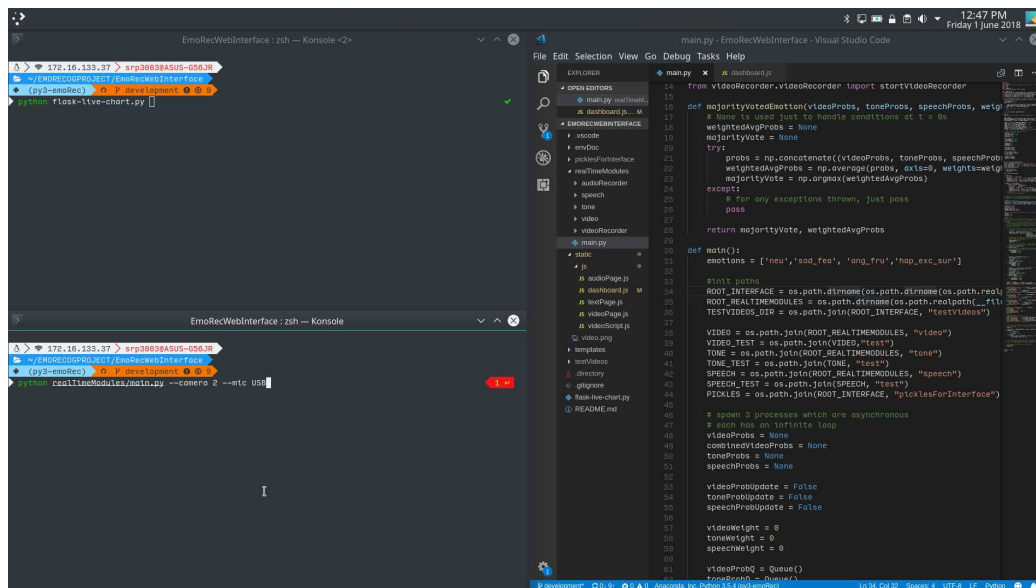


Figure 8.1: Project Execution

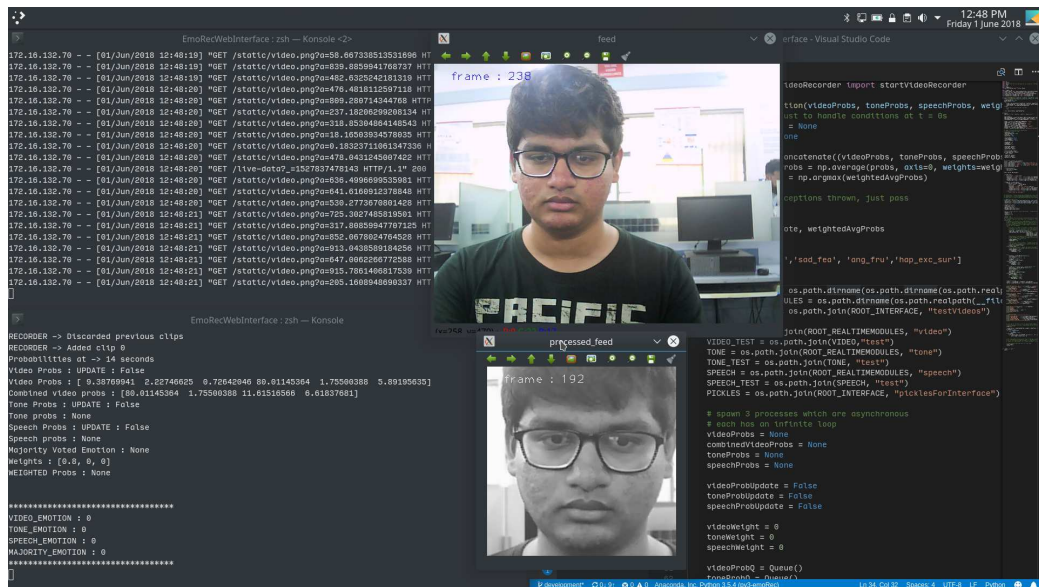


Figure 8.2: Neutral Emotion Recognition in Real Time

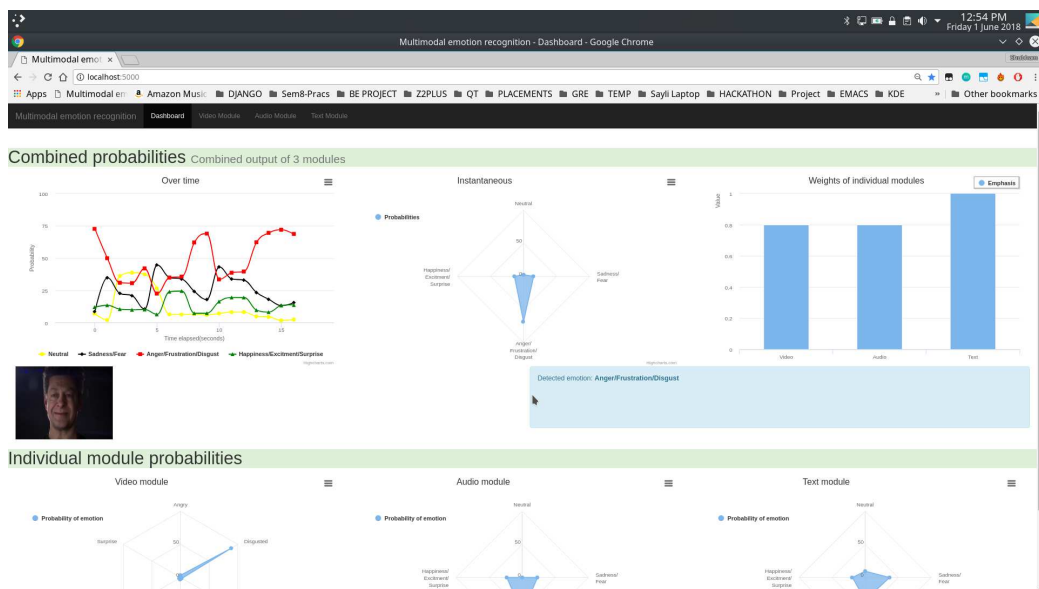


Figure 8.3: Anger Emotion Recognition in Real Time

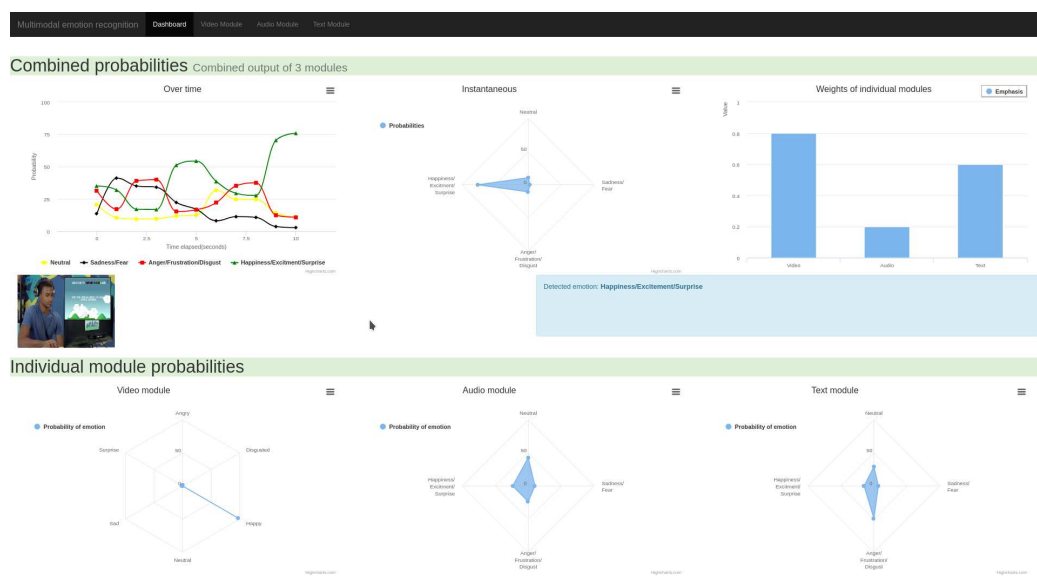


Figure 8.4: Happy Emotion Recognition in Real Time

Chapter 9

Deployment and Maintenance

9.1 INSTALLATION

- Ensure you have Python 3.6 installed on a 64-bit Linux distro.
- You will need a Google cloud API key for converting speech to text.
Get one from :

https://cloud.google.com/docs/authentication/api-keys#creating_an_api_key

- A brief description about the project structure:
 - envDoc : Contains all configuration files and dependencies.
 - realTimeModules : Contains implementations of each individual module and also a main process to spawn subprocesses for each module.
 - static : Contains static code for Flask webapp (mostly JavaScript scripts for real time rendering on webpage).
 - templates: Contains HTML webpages for client to view.
 - flask-live-chart.py : The Flask webapp.
 - realTimeModules/config.json : Config file to set configurable parameters. See the file comments for all the parameters which can be configured.

Project Structure:

Install dependencies:

```

EmotionWebInterface
├── envDoc
│   ├── env-conda-spec-FILE.txt
│   └── env-pip-packages.txt
├── Flask-Live-Chart.py
├── pickleInterface
│   └── pickleFile.py
├── README.md
├── realtimeModules
│   ├── audioRecorder
│   │   ├── __init__.py
│   │   ├── _pycache_
│   │   │   ├── __init__.cpython-35.pyc
│   │   │   ├── audioRecorder.cpython-35.pyc
│   │   │   ├── channel_index.cpython-35.pyc
│   │   │   └── __init__.cpython-35.pyc
│   │   └── test
│   ├── n2io.py
│   ├── speech
│   │   ├── api_key.json
│   │   ├── __init__.py
│   │   ├── _pycache_
│   │   │   ├── __init__.cpython-35.pyc
│   │   │   └── speech.cpython-35.pyc
│   │   ├── saveFile
│   │   ├── speech.py
│   │   └── transcriptions.txt
│   ├── tone
│   │   ├── defaultParameters.txt
│   │   ├── energy.py
│   │   ├── featureExtraction.py
│   │   ├── __init__.py
│   │   ├── _pycache_
│   │   │   ├── energy.cpython-35.pyc
│   │   │   ├── featureExtraction.cpython-35.pyc
│   │   │   ├── __init__.cpython-35.pyc
│   │   │   └── tone.cpython-35.pyc
│   │   ├── scalarParameters.txt
│   │   ├── test
│   │   └── tone.py
│   └── video
│       ├── emotionFaces1.xml
│       ├── haarCascade_FrontalFace_all2.xml
│       ├── haarCascade_FrontalFace_all_train.xml
│       ├── haarCascade_FrontalFace_all.xml
│       ├── haarCascade_FrontalFace_default.xml
│       ├── __init__.py
│       ├── _pycache_
│       │   ├── __init__.cpython-35.pyc
│       │   └── video.cpython-35.pyc
│       ├── shape_predictor_68_Face_Landmarks.dat
│       ├── test
│       │   └── videoPlayback.mp4
│       └── video.py
├── static
│   ├── js
│   │   ├── dashboard.js
│   │   └── videoScript.js
│   └── video.png
├── templates
│   ├── index.html
│   └── live-data.html
├── testVideos
├── audioEmotionEmotionExtractionFace.mp4
├── audioEmotionEmotionExtraction.mp4
├── audioEmotionExtraction.mp4
├── audioEmotionHappy.mp4

```

Figure 9.1: Project Structure

- statsmodels 8.0 :
Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator.
- pip 9.0.3:
To install python packages.
- Flask 12.2:
Flask is a micro web framework written in Python, used for hosting the web server.
- scikit-learn 19.1:
Machine learning library used for training and predicting.
- google-auth 1.4.1 :
Used for authenticating Google API key.
- google-api-core 1.1.2:
Dependency for google-cloud.
- google-cloud-speech 34.0:
Used for converging speech to text transcripts.
- matplotlib 2.0:
Used for plotting and visualizing data locally on the server.
- numpy 1.14.2:
Used for processing data vectors.
- pandas 22.0:
Used for quick data analysis.
- python-speech-features 6.0:
Used by text module.

- Optionally, you can create a virtual environment using the spec file given in /envDoc.

DEPLOYMENT :

- Clone the repository's master branch from <https://github.com/EmoRecog/EmoRecWebInterface.git>
- Switch to appropriate virtual environment with dependencies installed.
- Run `realTimeModules/main.py` to start all 3 subprocesses.
- Run `flask-live-chart.py` to start Flask server.
- You can now access the interface at $< IP >: 5000$ (5000 is the default port).

Chapter 10

Conclusion and future scope

- **Conclusion**

We have proposed an emotion recognition system comprising of three modules, tone, speech text and facial feature analysis. We tested the implementation of tone analysis model and speech text analysis model for emotion recognition with promising experimental results. The output emotive state of this emotion recognition system is to be used as reinforcement to determine a response using a neural responding machine. Thus implementing an emotion based reinforcement loop will make interaction with a robot more humane.

- **Future Scope**

- **Multimodal inputs from multiple channels** :: Algorithms currently designed assume that the signal per module is single channel. They can be extended to include multiple channels for a better affect-awareness module.
- **Using normalization per subject** :: The models described assume that no information about the subject is available prior to exposure of the model to the subject. The collected information over a period of time can be used for user adaptation and personalization, for example, to implement gesture dynamics.
- **Facial gaze tracking** :: Along with facial features, gaze tracking can be implemented to improve emotion recognition.
- **Heirarchical training** :: In the tone analysis module, 10% of the samples with highest energy are utilized for training DNN. Heirarchical learning can be implemented to choose best training samples directly. (Apply training sub-datasets to a sequence of

DNNs, during testing stage, the input sample is simultaneously applied to all of these DNNs and decision is aggregated).

- **Incorporate temporal dynamics** :: Speed at which a transition can take place is termed as temporal dynamics of a system. The change in emotional state can be studied and incorporated into the recognition mechanism.
- **Learning from spectrograms** :: Instead of using handcrafted features like MFCC, it should be possible to learn emotional information from spectrogram with techniques like Mel-filter bank.

Annexure A

Laboratory assignments on Project Analysis of Algorithmic Design

- To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.

<20>	<20>	<20>	<20>
I	D	E	A
Increase	Drive	Enhance	Accelerate
Accuracy of emotion detection and recognition.	Create a system which can recognize an emotion based on facial cues and voice samples.	Images of faces and audio dataset is used and classifiers like RNN for audio, SVM for speech and CNN for facial.	Speed of emotion detection and recognition.
Improve	Deliver	Evaluate	Associate
The ability of the system to classify various emotions by combining facial cues with voice samples.	An emotion recognition system based on principles of image processing, audio processing and machine learning.	Images in the form of sequential frames and voice samples in the training datasets to classify emotions correctly.	The received live voice sample and frame sequence to the information stored in the system.
Ignore	Decrease	Eliminate	Avoid
Irrelevant audio and background images in the dataset as well as the live feed.	Delay in the audio and image transmission from NAO to the remote server.	Fault in the emotion recognition as well as the delay in classification.	Errors which may arise due to manual or background interference.

Figure A.1: Idea Matrix

- Analysis of algorithms and mathematical modelling.

- **Need for Analysis to be done:**

Informally an algorithm is any well-defined computational procedure that takes some value or a set of values as input and produces some value, or set of values as output. An algorithm is thus a sequence of computational steps that transform the input into output. To analyse an algorithm is to determine the amount of resources (such as time and storage) necessary to execute it. In order to choose the best algorithm for a particular task, we need to be able to judge how long a particular solution will take to run; or how long two solutions will take to run and choose the better of the two. We don't need to know how many minutes and seconds they will take, but we need some way to compare algorithms against one another. This is why we need to analyse an algorithm.

- **Problem Complexity:**

1. **P:**

If the running time is some polynomial function of the size of input, for instance if the algorithm runs in linear time or quadratic time or cubic time, then we say that algorithm runs in polynomial time and the problem solves in class P.

2. **NP:**

There are a lot of programs that don't run in polynomial time on regular computer, but do run in polynomial time on nondeterministic Turing machine. These programs solve the problems in NP, which stands for nondeterministic polynomial time. NP problems are solvable in nondeterministic polynomial time, but variable in deterministic polynomial time.

3. **NP-Complete:**

NP is a complexity class which represents the set of all problems X for which it is possible to reduce any other NP problem Y to X in polynomial time. Intuitively this means that we can solve Y quickly if we know how to solve X quickly. Precisely, Y is reducible to X, if there is a polynomial time algorithm f to transform instances y of Y to instances $x = f(y)$ of X in

polynomial time, with the property that the answer to y is yes, if and only if the answer to $f(y)$ is yes. NP-complete problems are atleast as hard as all the other NP problems. Levin-Cooks theorem gives that CNF-satisfiability problem is NP complete.

4. NP-Hard:

These problems are the problems that are even harder than the NP-complete problems. NP-hard problems do not have to be in NP, and they do not have to be decision problems. The precise denition here is that a problem X is NP-hard, if there is an NP-complete problem Y , such that Y is reducible to X in polynomial time. But since any NP-complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

– Mathematical Model:

* Tone Analysis:

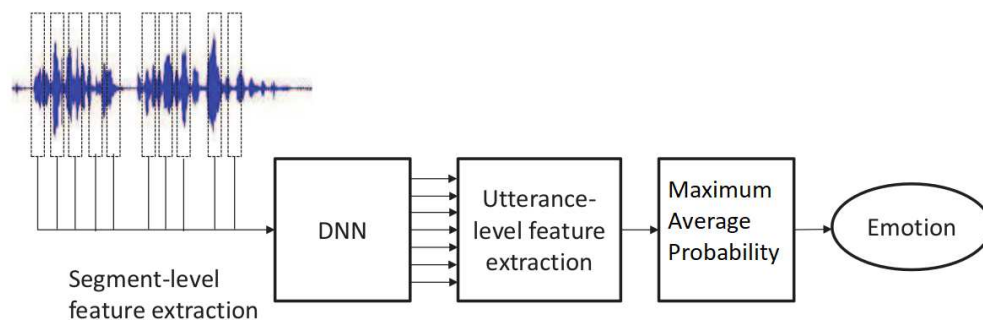


Figure A.2: Tone Analysis

1. Input to DNN:

- (a) Frames composed into segments according a sliding

window, certain top percentage of segments qualify as input with respect to their energy.

- (b) MFCC features, pitch based features are extracted per frame to give feature vector per frame $z(m)$
- (c) $2m+1$ frames are stacked to generate a segment level feature vector $x(m)$

$$x(m) = [z(m-w), \dots, z(m), \dots, z(m+w)] \quad (\text{A.1})$$

2. Output from DNN:

A sequence of probability distribution t over all emotion states for each segment

$$t = [P(E_1), \dots, P(E_k)]^T \quad (\text{A.2})$$

3. Input to ELM (Utterance level feature extraction):

Statistical features per each probability distribution.

f_1, f_2, f_3 which correspond to maximal, minimal and mean of segmentlevel probability of k th level emotion over utterance.

f_4 is percentage of the segments which have high probability of emotion k .

4. Output of ELM(Utterance level classier):

K -dimensional vector corresponding to scores of each emotion state. (k emotions considered).

5. Objective function for DNN: Gradient descent - mini-batch

6. Objective function for DNN: Cross entropy

7. ELM is trained with least squared error.

* Speech Analysis:

· Training Stage:

Input: Dataset split into training set and testing(validation) set with k -fold cross-validation for assessing accuracy.

Output: Trained classier with Likelihood

Likelihood(evidence) calculation: Conditional probabilities of attributes for class labels are calculated from the dataset, termed as evidence Z .

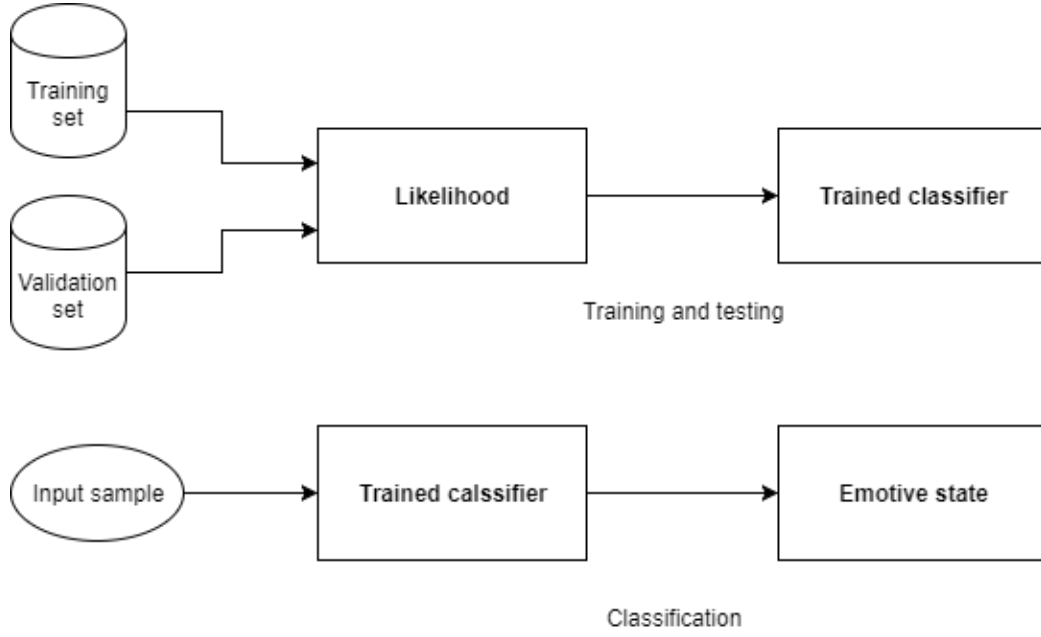


Figure A.3: Speech Analysis

$$Z = p(x) = \sum_k p(C_k)p(x|C_k) \quad (\text{A.3})$$

· **Classification Stage:**

This stage uses the likelihood or evidence calculated in training to classify a novel input.

$$P(c|d) \propto P(c)\Pi_{1knd}P(tk|c) \quad (\text{A.4})$$

In text classication, the goal is to nd the best class for the input text. The best class in Multinomial NB is the most likely or maximum posteriori (MAP) class c_{map}

$$c_{map} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(c)\Pi_{1knd}P(t_k|c) \quad (\text{A.5})$$

In the above equation, many conditional probabilites are multiplied, and with a large enough vocabulary, raw multiplication will almost denitely result in an underow. It is therefore better to perform the computation by adding logarithms of probabilites instead of multiplying probabilites. The class with highest log probability score

is still the most probable; $\log(xy) = \log(x) + \log(y)$ and the logarithmic function is monotonic. Hence, the maximization that is actually done in our implementation of the Multinomial NB classier is as follows:

$$c_{map} = \underset{c \in C}{\operatorname{argmax}} [\log P(c) + \sum_{1 \leq k \leq d} \log P(t_k | c)] \quad (\text{A.6})$$

* **Facial Analysis Module:**

- **Input:** Image (frame) from the video

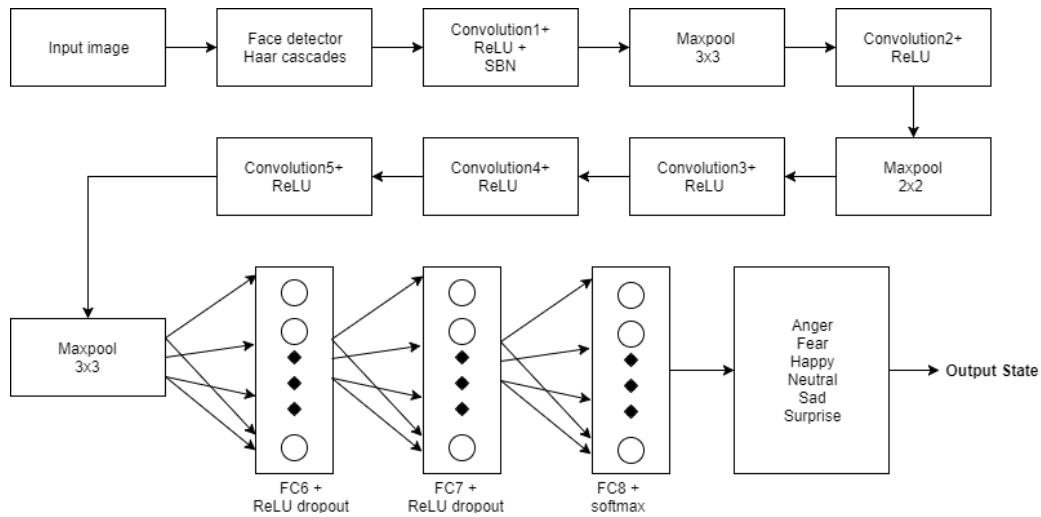


Figure A.4: Facial Analysis

- **Output:** Prediction based on the output softmax layer
- **Process:**

Image processing is narrowed to the regions of the image containing faces, by performing facial recognition in the image using a classier (Haars cascades)

Convolutional Layer: It computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume

ReLU layer: It applies element-wise activation function which is rectier function = $\max(0, x)$ thresholding at zero.

Pool layer: It performs downsampling operation along spatial dimensions.

Fully-Connected (FC) layer: It computes the class scores among the n categories. Each neuron in this layer is connected to all the numbers in the previous one. Dropout randomly ignores nodes to prevent interdependencies emerging between nodes.

* **Response Generation:**

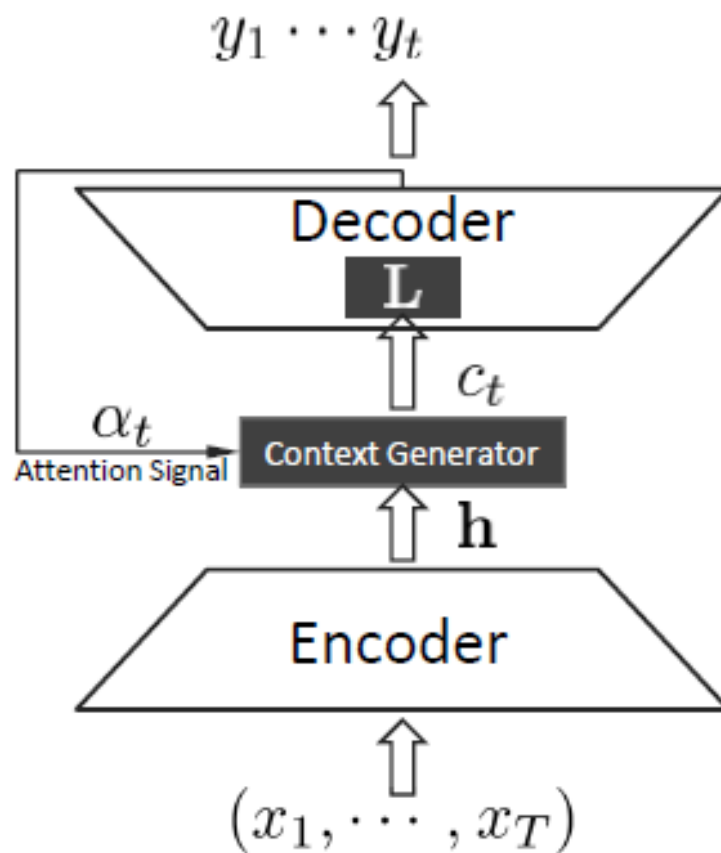


Figure A.5: Response Generation

- Basic idea of NRM is to build a hidden representation of a statement, then generate a response based on it.
- Encoder converts input sequence (x_1, \dots, x_T) into a set of highdimensional hidden representations $h = (h_1, \dots, h_T)$

- h and attention signal (previous decoded response) t are fed into context generator to build context input to decoder ct .
- ct is linearly transformed by matrix L (as a part of the decoder) into a stimulus of generating RNN to produce the t -th word of a response yt .

– **Algorithm Type:**

- * Backpropagation is a statistical method for function approximation using multilayer neural networks. Backpropagation is performed with gradient descent function.
- * Backpropagation algorithm has the disadvantage that it becomes very slow in at regions of error function. In that case the algorithm should use a larger iteration step. However, this is precluded by the length of the gradient, which is too small in these problematic regions. Gradient descent can be slowed arbitrarily in these cases.
- * It is proved that finding the appropriate weights for a learning problem consisting of just one input-output pair is computationally hard, by mapping it to an 3SAT NP complete problem. (Hilberts problem and Kolmogorovs theorem)
- * Thus, backpropagation algorithm is NP complete, no polynomial time algorithm is known.

– **Feasibility Study:**

1. **Technical Feasibility:** Required hardware resources for training classifiers are available.
Licensed softwares and libraries are available.
2. **Economic Feasibility:** Datasets and libraries used are provided under licenses which allow free use for non-commercial purposes.
3. **Time Feasibility:** The training of deep neural networks largely constitutes the time requirements for the projects involving deep learning.
4. **Privacy Feasibility:** Datasets and libraries are provided with licenses which allow use for non-commercial purposes. GNU-GPL license, MIT license and Apache license allow use of datasets and libraries for non-commercial purposes. GNU GPL allows free use and distribution of software under

its license, as long as its derivatives follow the same licensing model.

- **Parallelism in the project:**

- A hierarchy of flow of control among the modules is shown in the figure A.6
- Modules at the same level in the hierarchy are functionally independent.
- Data dependencies, if they emerge among the modules can be handled by mutually exclusive access or by increasing redundancy of data, making the replicated data private to each thread.
- Mutually exclusive access can be enforced with thread safe functions or by implementing monitors.

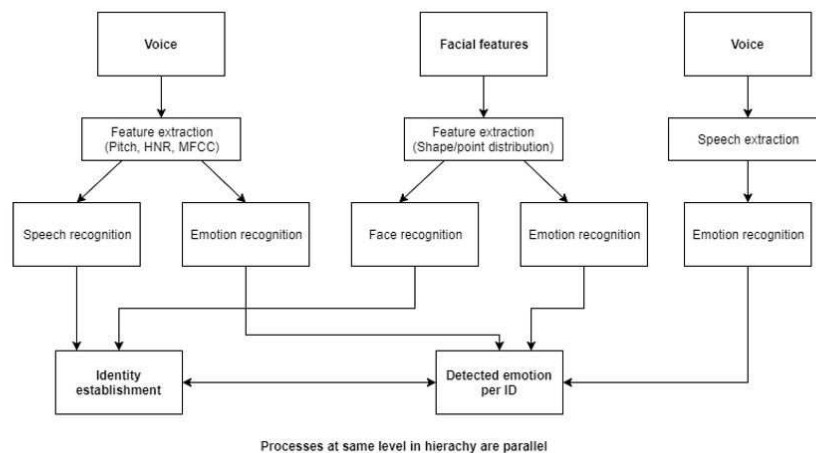


Figure A.6: Parallelism in the project

- Increasing redundancy of data can result in inconsistent states, if the data is mutable. Redundancy is to be implemented only for immutable data, that is only to be read, such as audio and video frames.
- Modules at the next level of hierarchy are blocked until all the modules at the current level have finished execution.

- Thus modules are functionally dependent at consecutive levels.
- Feature extraction is common to identity recognition and emotion recognition tasks. For voice, handcrafted features which are extracted such as pitch characteristics, Harmonics to Noise Ratio, MelFrequency Cepstral Coefficients are required for both identity recognition and emotion recognition using speech. Similarly for facial features, extracted features are common for identity and emotional recognition tasks.
- Identity and emotion recognition tasks in each module can be performed parallelly across the modules as they are functionally independent.
- Establishment of identity using outputs from all modules and linking recognised emotion state is performed at the next level. This level cannot be processed until all the tasks in the previous level have concluded. Thus level-to-level now has functional dependency.

Annexure B

Term-II Project Laboratory Assignments

1. Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment

- Our project in its essence contains three independent modules covering the three key contributors which give away the emotion a person is feeling. The contributors selected by us were the facial cues, audio, and textual content. We demonstrated the first speech module, which took sample statements from Nao robot and analyzed them to produce a discrete emotion label. In the mid term assessment, we presented the second module which took facial cues and predicted a discrete emotion label in real-time. We also demonstrated the text content module. Following are the suggestions received from the assessments :

– **Green coding:**

We were instructed not to implement every single functionality ourselves, if a certain minor functionality was already available in open-source APIs (speech recognition before speech emotion recognition), we were instructed to use those. The amount spent for productive programming was to be increased.

We used APIs for MFCC calculation, facial landmark detection and speech-to-text.

MFCC - For calculation of energy of audio segments in a sample, calculation of Mel Frequency Cepstral Coefficients per segment.

Facial landmark detection - dlib (allows tracking of 68 facial

landmarks)
Speech-to-text - Google S2T

– **Consistent coding styles:**

We were instructed in the mid-semester assesment to use coding styles consisten with other teammates. To that end, we have used auotpep8 which provides a PEP 8 style guide. Pylint extension for IDEs has been used

- **Emphasis on GUI:** Our models previously gave the outputs on command-line, in the form text output for the discrete emotion labe, with the corresponding confidence score and any other exceptions thrown ("No face detected" for example, in case of the visual model). We were instructed to make the front-end interace visually appealing which summarized the data in form of reporting methods such as graphs. The idea was to make it more presentable for a non-technical demographic.
We plan to use python-django binding for a web interface, with reporting packages : matplotlib and seaborn. We plan to make the web pages reactive and real-time.

- **Maintaing continuous data reports:** We were instructed to store our training graphs and script outputs for intermediate processing. We have stored the data segretation scripts, data cleaning scripts and training results.

• **Programming of project functions, interfaces and GUI**

– **Speech Analysis Module:**

(a) **Function:**

Deep neural network to generate statistical attributes for segements of a the audio sample, Extreme learning machine to predict an emotion label for the audio sample from the high-level statistical functions provided by DNN. In case of DNN, all segements of the audio assume the same emotion label.

(b) **Interface:**

Microphone source, either from the local machine (testing and web interface) or Nao Robot - Single channel source sampled at 16Khz. Audio samples are retrieved from Nao Robot via SCP protocol from the paramiko package.

(c) **Output:**

Output is a discrete emotion label.

– **Face Analysis Module:**

(a) **Function:**

68 landmarks are detected on the face under consideration. Then distances of each landmark are calculated from the cumulative centroid, which are used as attributes for the prediction model, polynomial kernel SVM. We have handled tilting of the face with rotational transposition.

(b) **Input:**

Video source, either from the local machine (webcam, for testing and web interface) and Nao robot front camera.

(c) **Output:**

Realtime graph plots of the detected emotion per frame. Frame skip is a configurable parameter for real-time processing, according to available computational capabilities.

A web interface is to be used for presenting the outputs from each of the module independently, as well as a combined output from the three modules.

Interface - Microphone source, either from the local machine (testing and web interface) or Nao Robot - Single channel source sampled at 16Khz. Audio samples are retrieved from Nao Robot via SCP protocol from the paramiko package. Output - Output is a discrete emotion label.

2. Project workstation selection, installations along with setup and installation report preparations

- **Workstation:**

Based on the minimum requirements for our software stack, our projects workstation should have:

- (a) 64 or 32 bit Intel i3 processor or higher
- (b) 4 GB RAM
- (c) Ubuntu 16.04
- (d) NVIDIA Graphics card

- **Software stack:**

- (a) **Anaconda:** Free and open source package used for maintaining separate environments for Python programming language.

Installation Guide:

- i. Download installer from website.
 - ii. In a terminal, run:
 - iii. *bashAnaconda – latest – Linux – x86_64.sh*
 - iv. Follow prompts and install.
 - v. **To create a virtual environment:**

Open terminal and run command :

`conda create -n yourenvname python=x.x anaconda`

- (b) **NAO SDK:** Python/C++ bindings to interact with NAO bot. We use this to get input and make the bot perform actions as per detected emotion.

Installation:

- i. Download installer file from website.
 - ii. In a terminal, run:
`pynaoqi-python-2.7-naoqi-x.x-linux32.tar.gz`
 - iii. Set the PYTHONPATH variable:

`export PYTHONPATH=${PYTHONPATH}:/path/to/python-sdk`

- **Installation:**

OpenCV: Image processing library used for video module.

Installation:

- (a) Download repository:
`git clone https://github.com/opencv/opencv.git`
`cd opencv`

```

git checkout 3.3.1
cd ..
(b) Create build directory:
    cd opencv
    mkdir build
    cd build
    Run cmake to configure compilation flags:
cmake -D CMAKE_BUILD_TYPE=RELEASE

-D CMAKE_INSTALL_PREFIX=/usr/local

-D INSTALL_C_EXAMPLES=ON

-D INSTALL_PYTHON_EXAMPLES=ON

-D WITH_TBB=ON

-D WITH_V4L=ON

-D WITH_QT=ON

-D WITH_OPENGL=ON

-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules

-D BUILD_EXAMPLES=ON ..

Compile and install:

make -j4

sudo make install

sudo sh -c 'echo "/usr/local/lib" >> /etc/ld.so.conf.d/opencv.conf'

sudo ldconfig

```

- **Scikit-learn:** Machine learning library in Python used to create and train various classifiers.

Installation:

- (a) Switch to virtual environment where you want to install:

```
source activate yourenvname
```

(b) Install scikit-learn:

```
conda install scikit-learn
```

- **Nltk:** Natural Language Toolkit - Used for text and speech processing.

Installation:

```
sudo pip install -U nltk
```


Annexure C

Information of Project Group Members

C.1



1. **Name :** Manas Kale
2. **Date of Birth :** 01/03/1999
3. **Gender :** Male
4. **Permanent Address :** [REDACTED]
[REDACTED]
5. **E-Mail :** [REDACTED]
6. **Mobile/Contact No. :** [REDACTED]
[REDACTED]

C.2



1. **Name** : Shubham Punekar

2. **Date of Birth** : [REDACTED]

3. **Gender** : Male

4. **Permanent Address** : [REDACTED]
[REDACTED]

5. **E-Mail** : [REDACTED]

6. **Mobile/Contact No.** : [REDACTED]
[REDACTED]

C.3



1. Name : Rohit Patankar

2. Date of Birth : [REDACTED]

3. Gender : Male

4. Permanent Address : [REDACTED]
[REDACTED]

5. E-Mail : [REDACTED]

6. Mobile/Contact No. : [REDACTED]
[REDACTED]

C.4



1. **Name** : Saurabh Shirodkar

2. **Date of Birth** : [REDACTED]

3. **Gender** : Male

4. **Permanent Address** : [REDACTED]
[REDACTED]

5. **E-Mail** : [REDACTED]

6. **Mobile/Contact No.** : [REDACTED]

[REDACTED]