E-MO: Emotional Management Object

Authors: Patrick Gao, Vinay Kukutla, Yoojin Lee: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of analyzing user sentiment through facial and text analysis. E-MO is a robot therapist that analyzes the user's emotion by capturing their face and speech and performing facial emotion recognition and sentiment analysis. It also acts as an interactive diary, storing the user's verbal journal entries as text in a web application labeled by emotion. E-MO provides instant therapeutic feedback to the user's environment by adjusting IoT devices such as lighting, temperature, and music connected devices.**

***Index Terms* — computer vision, facial emotion recognition, sentiment analysis, tone recognition, mental health**

I. Introduction

Methods to treat mental illness today are relatively archaic compared to the current technology available. Mental health experts are unable to monitor patients day-to-day unless they are confined to daily monitoring in an institution. We aimed to create a robot therapist that can provide daily monitoring of a patient's improvement at home through casual conversation interactions. The robot uses a combination of sentiment analysis, tone recognition, and facial emotion recognition to analyze the user's mood and respond by creating a fitting environment using IoT devices such as speakers, lights, and thermostats. We used speech-to-text to transcribe the user's conversation and use sentiment analysis to extract an emotion label from the text. We used Google Cloud speech-to-text API for its high accuracy and immense amount of training that we could not possibly match with our own proprietary speech-to-text algorithm. For facial emotion recognition, we used a pre-trained CNN to extract features from the user's face, which we then classify into one of six emotions with our trained SVM. These emotions include neutral, happiness, sadness, anger, surprise, disgust, and fear. We trained the sentiment analysis using the Sentiment140 dataset and further text samples scraped from the web. Tone recognition was trained on the Ryerson Audio-Visual Database and yields an output of either positive or negative. The facial emotion recognition SVM was trained with the commonly used CK+[5] and AffectNet[4] datasets. These three evaluations of the user's emotion are combined to make a final emotion classification which is used to decide what kind of environment to create for the user. We planned to use the Spotify API to select the appropriate songs to play and smart LED bulbs to set lighting. Spotify has hundreds of playlists, sorted by mood, that can be simply matched with the user's emotion. The sentiment analysis, tone recognition, and facial emotion recognition algorithms were to be hosted on an AWS instance, with the result being sent back to a Raspberry Pi on the robot. We planned to create the body of the robot through 3D printing and laser cutting. This body would have housed a camera, microphone, and screen. Unfortunately, the COVID-19 outbreak has cut short our plans to implement IoT device responses and the physical portion of the robot was removed. More detailed refocus steps are outlined in section *VII.E*.

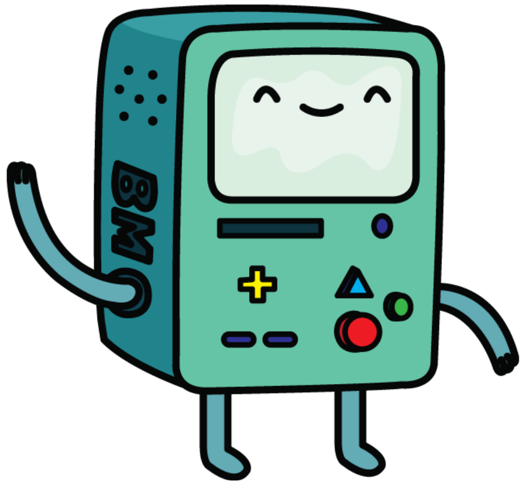


Figure 1: *B-MO*, a character from *Adventure Time* - our inspiration for our physical design.

II. Design Requirements

Users of E-MO are expected to be sitting squarely in front of E-MO, approximately within 6 ft of E-MO's camera. High degrees of face angles drastically reduce the ability to detect the face and perform facial emotion recognition, as such we expect the user to be directly facing E-MO. E-MO also only expects a single user at a time. Current facial emotion recognition technology peaks at around 70-85% accuracy, although real world performance typically had around 50% accuracy. We aimed for 65% accuracy as our use case involves fewer face angles which are the main detriment to accuracy. Sentiment analysis also peaks at 70-85% accuracy. We aimed for 65% accuracy since our text is transcribed and lacks major sentiment signs such as emoticons and punctuation. Tone recognition has not been as thoroughly researched, with some papers claiming up to 80% accuracy on binary classification but much lower numbers on 7 emotion classification. We aimed for 60% accuracy to match the performance of our other two classifiers.

Under the assumption that the three methods of emotion detection will constructively improve our classification accuracy, we aimed to have an accuracy of 75% overall. We hoped to analyze the user's face and take action in under 10 seconds assuming a stable connection to the internet.

There are hardware requirements for our project to ensure that the data we capture can be used effectively and that all the components can run efficiently. The camera used must be able to capture 224x224px images as that is the input size required of our facial emotion recognition algorithm. We chose a 1080p camera at 30fps as these specs cover this requirement. The microphone should be able to meet our required sampling rate of 16khz for our sentiment analysis process.

Our MVP was be a device with a microphone and a camera that can detect your emotion based on your speech and facial expressions. This requires the successful integration of the three trained networks into our product. The device must also successfully communicate with the server, sending data to be computed and receiving the response. Our server should be able to use both the facial emotion and speech data to compute the emotion with 75% accuracy.

III. Architecture

Our system is divided into 4 parts:

- ❏ Facial emotion recognition
- ❏ Sentiment analysis
- ❏ Voice tone analysis
- ❏ Web application

The overall architecture diagram can be found in Figure 7. The user inputs to the system are video and audio, where frames are captured from the video every second for FER and audio is divided into sentence chunks for tone recognition and sentiment analysis. Audio is also converted to text and sent to the web application through HTTP requests to be stored in the database along with the final emotion label. The user can later access their journal entries through the web application. The specific details for each component are discussed in section VI.

IV. Principle of Operation

E-MO expects a single user to be sitting directly opposing it within arms reach. Facial detection is performed using the OpenCV frontal face Haar cascade. High degrees of face angles drastically reduce the ability to detect the face and perform facial emotion recognition, and as such we expect the user to be directly facing E-MO. To gauge the distance of the user, the size of the bounding box capturing the user's face must be above a certain threshold for E-MO to initiate a prompt and begin listening and performing facial emotion recognition. If multiple faces are detected in frame, the largest bounding box will be used for engaging E-MO and for facial emotion recognition, eliminating unintentional interference of people in the background. We can rely on Google's built-in background noise filtering in their speech-to-text API to eliminate the effect of other people speaking around the user. On the web application, users are able to navigate to the new entry page to begin a new recording. After beginning the recording, we activate both facial emotion recognition and Google's speech-to-text. Facial emotion recognition is performed at a rate of 1/sec for the duration of the user's speech. The most frequent emotion classification is used as the final facial emotion label. Once the user finishes speaking, audio is divided into sentences and sent to Google's speech-to-text API, after which we receive a transcribed text for us to perform sentiment analysis. The divided audio is also used for tone recognition to produce a positive/negative label. The recording along with our final emotion label is sent to our web application to be stored.

All following features were planned before our project refocus and were therefore not implemented. After the emotion is analyzed, the Raspberry Pi will send out requests to different IoT devices with the requests based on the type of emotion the user is feeling. Using Spotify, Philips Hue and Nest APIs, we control the user's environment from the Raspberry Pi. As long as the user is in front of E-MO, IoT devices will change every 5 minutes to suit their current mood. Different colors, music and temperatures affect human emotion in different ways. E-MO can leverage these factors to provide a user emotional relief and therapy.

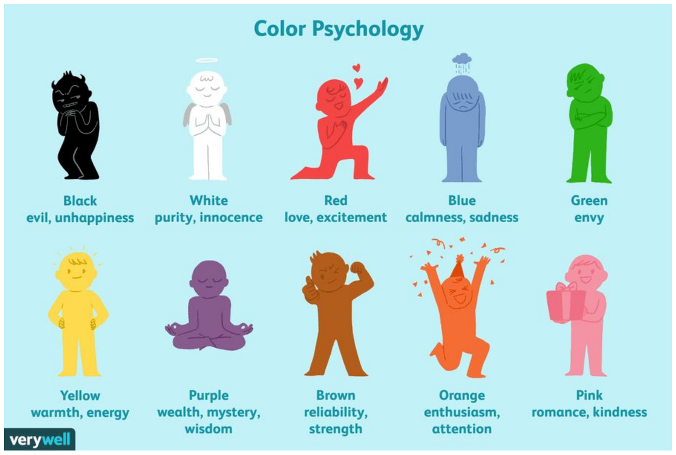


Figure 2. Colors and their corresponding emotions (Cherry).

V. Design Trade Studies

The trade studies for our project are described in the individual sub-system descriptions in section VI. In summary, we considered various approaches to training and the use of different algorithms for emotion analysis, optimizing for highest accuracy.

VI. System Description

A. *Facial Emotion Recognition (FER)*

There are several widely used methods for facial emotion recognition including deep networks with end-to-end learning and convolutional neural networks with long short-term memory (LSTM) for frame-to-frame temporal features. The obvious advantage of these complex network architectures is highly fine-tuned extraction and processing of facial features for facial emotion recognition. Deep networks with end-to-end learning skips facial landmark extraction and enables learning straight from input images, while LSTMs allow for temporal feature tracking in sequences that have been shown to have higher accuracy than FER on static images. However, the downside to deep networks is the need for very large amounts of training data to achieve improved performance compared to standard CNNs. We overcome this by following a transfer learning approach[9], in which all but the final fully-connected layers of a pre-trained deep network are used to extract features to pass into our own classification SVM. LSTMs also offered promising advantages to our use case as we use video input for our facial emotion recognition, but some shortcomings to LSTMs are the potential for facial landmarks between sequences to become lost during image normalization and variations in temporal scale in facial expressions among different people. As a result, we did not use LSTMs for our facial emotion recognition.

1. Image Feature Extraction

The VGG19 image classification network was used for image feature extraction. This network is used to classify up to 1,000 labels and trained on 1.3 million images. The network architecture is 19 layers deep with 5 max-pooling layers and 3 fully-connected layers at the end as shown in Table I. All other layers are convolutional layers using a 3x3 convolution with 1 padding and 1 stride. Inputs are 224x224 pixel images. More information about the architecture and training/testing methods can be found in the reference paper[5]. The paper we referenced for the transfer learning approach[9] performed an assessment of which max pooling layers provided the highest accuracy for facial emotion recognition. From Table II, we see that the 4th max-pooling layer led to the highest training and test accuracy on the CK+ and JAFFE datasets. Therefore we use the feature vector from the 4th pooling layer as input to our SVM. This feature vector is size (1, 14, 14, 512), giving a total of 100,352 data points. The transfer learning approach utilized principal component analysis at this stage to improve processing speed, to an local optimum of 200 features among the set of {50, 100, 150, 200}. Our timing requirements are less strict and therefore we skip PCA for our algorithm.

Table I. VGG19 network architecture (9).

Layer (type)	Output Shape	Parameters
input_1 (Input Layer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	**(None, 112, 112, 64)**	**0**
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	**(None, 56, 56, 128)**	**0**
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	**(None, 28, 28, 256)**	**0**
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	**(None, 14, 14, 512)**	**0**
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	**(None, 7, 7, 512)**	**0**
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	**(None, 4096)**	**102764544**
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

2. SVM Classifier

Our SVM is built on the Sci-kit Learn library, using its SVC classifier for multi-class support vector classification into 7 emotion labels. Multi-class classification is implemented with a one-vs-all approach although a one-vs-one approach can be further tested. Preliminary training was performed on the Cohn-Kanade Extended (CK+)[8] dataset, using the peak expression in each subject's sequence of images for static emotion classification. These images are labeled with a number 0-6 for each of the 7 emotion classifications as shown in Table III.

Table II. VGG19 pooling layer assessment (9).

	CK+ Dataset			JAFFE Dataset		
Features	PCA - 100			PCA – 200		
VGG19 Layer (POOL)	Training (80%) 10-Fold	Training (80%) A_{JK}	Test (20%)	Training (80%) 10-Fold	Training (80%) A_{JK}	Test (20%)
block1	87.90	88.69	85.71	77.27	79.52	88.10
block2	90.27	90.48	85.71	81.66	83.73	88.10
block3	91.51	93.45	90.48	90.47	89.76	92.86
block4	**94.93**	**92.26**	**92.86**	**92.70**	**92.77**	**92.86**
block5	88.83	90.48	90.48	81.35	82.53	85.71
(Dense) FC1	91.76	89.88	92.86	81.01	76.51	88.10

Initial testing with 327 images from the CK+ dataset yielded around 50% testing accuracy using an 8-2 train-test split, but real-world performance was lacking. No quantifiable measurements were taken for real-world performance but a test program using live camera input was unsatisfactory for our final requirements. We predict that this is likely due to the small sample size of the CK+ dataset resulting in poor performance in non-laboratory conditions where variation in face angle and expression negatively affect accuracy. We move on to the AffectNet[7] database which sources 450,000 images from the Internet to train our model.

Table III. Numbered Emotion labels

0	Neutral
1	Happy
2	Sad
3	Surprise
4	Fear
5	Disgust
6	Anger

Using the manually annotated images from the AffectNet database, initial training used 60,000 images with an 8-2 train-test split yielding about 60% accuracy. Testing showed high accuracy for *happy* but all other emotions were lacking. Reviewing the training data revealed that a disproportionate number of training images were labeled as *happy*, thereby skewing our model. Balancing the training data among all emotions improved testing accuracy by ~3% overall, but not as significantly as we had hoped. Training with multiple epochs over the data also improved accuracy by another ~3%, but analysis of the confusion matrix showed that *fear, anger, disgust,* and *sad* suffered in accuracy compared to *happy*, *neutral*, and *surprised*. Adding a secondary SVM for the lower-performing emotion labels again improved accuracy by another couple percent. Looking at the confusion matrix more carefully, we see that *neutral/sad*, *surprise/fear*, and *anger/disgust* were frequently confused. This leads to our final SVM structure as shown in Figure 3. With this model, accuracy improved significantly across all emotions, yielding a final overall accuracy of 86.9%. Individual emotion accuracies are shown in Figure 4.

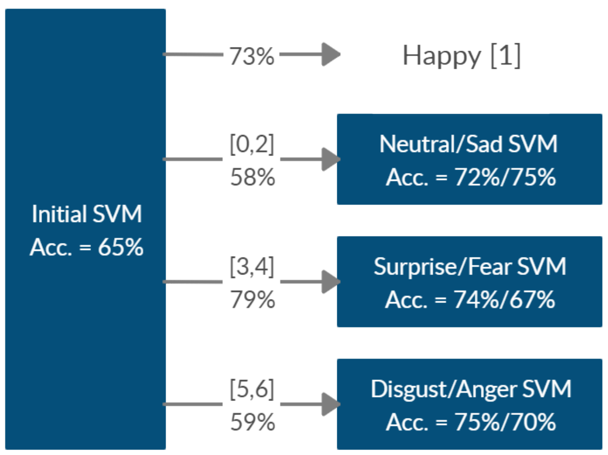
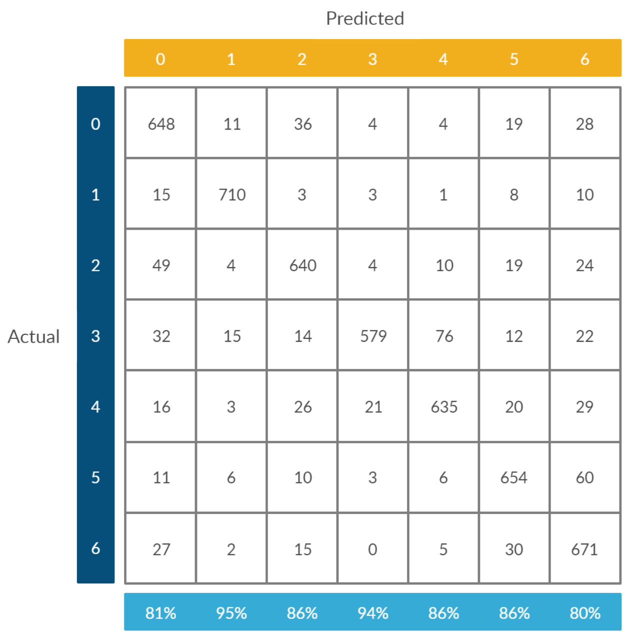


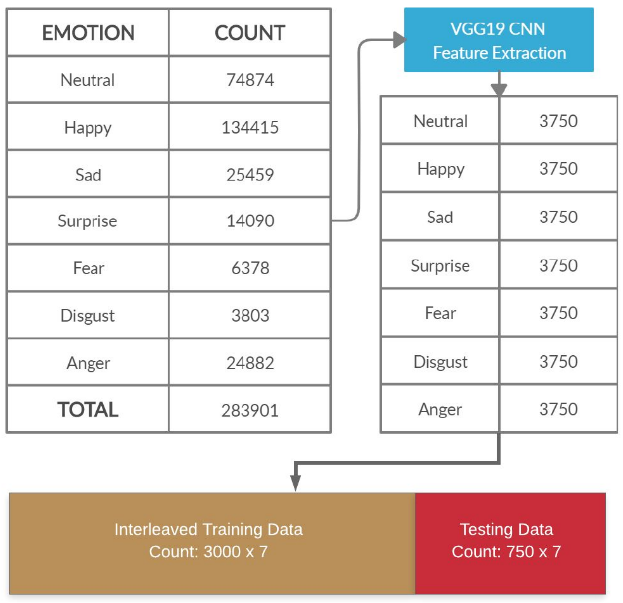
Figure 3. SVM structure.



Actual \ Predicted	0	1	2	3	4	5	6
0	648	11	36	4	4	19	28
1	15	710	3	3	1	8	10
2	49	4	640	4	10	19	24
3	32	15	14	579	76	12	22
4	16	3	26	21	635	20	29
5	11	6	10	3	6	654	60
6	27	2	15	0	5	30	671
	81%	95%	86%	94%	86%	86%	80%

Figure 4. SVM confusion matrix.

Our final training pipeline is depicted in Figure 5, using 3,750 images for each emotion. Each emotion is divided using an 8-2 train-test split with the training images interleaved by emotion. A step of 2,100 was used over 10 epochs of training data, and validation was performed using the same live-camera-input test program as mentioned previously.



EMOTION	COUNT
Neutral	74874
Happy	134415
Sad	25459
Surprise	14090
Fear	6378
Disgust	3803
Anger	24882
TOTAL	283901

Neutral	3750
Happy	3750
Sad	3750
Surprise	3750
Fear	3750
Disgust	3750
Anger	3750

VGG19 CNN Feature Extraction
Interleaved Training Data Count: 3000 x 7
Testing Data Count: 750 x 7

Figure 5. Training data processing pipeline.

B. Sentiment Analysis

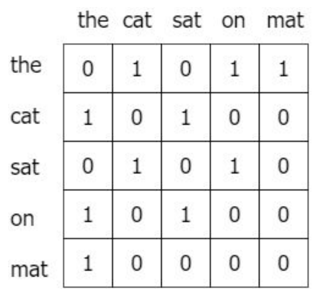
The second form of emotion analysis involves analyzing the text of the journal entry itself. Since neural networks require numbers as input data, raw text must be converted before it can be analyzed. The process of converting text to a numerical form is called vectorization. The most popular approach to vectorize words is the bag of words model. Sci-Kit Learn's CountVectorizer and TF-IDF Vectorizer are commonly used to do this. The CountVectorizer uses the basic approach of one-hot encoding. If the word appears in a sequence, its position in the vector is updated to a 1, else it is left as 0. A TD-IDF or term frequency-inverse document frequency vectorizer is usually a more accurate vectorizer as it factors in the length of the sequence and rarity of the word across the collection of corpus

$$TF(t) = \frac{\textit{Number of times term t appears in a document}}{\textit{Total number of terms in the document}} \qquad \text{(I)}$$

$$IDF(t) = log_e\left(\frac{\textit{Total number of documents}}{\textit{Number of documents with term t in it}}\right) \qquad \text{(II)}$$

Equations I, II. Term Frequency (I) and Inverse Document Frequency (II).

However, most vectorizers rely at most on local information only to characterize a word. This does not work well if a sentence has insignificant words surrounding it. This is what Stanford NLP's GloVe or Global Vectors for Word Representation addresses. GloVe uses a co-occurrence matrix for words, counting how many times a word *x* has co-occurred with a word *y*.



	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

Figure 6. A word co-occurrence matrix

For example, let's say we have three words i, j and k with i = 'dog' and j = 'human'. P_{ij} denotes the probability of the seeing words i (dog) and j (human) together. This is calculated by taking the occurrences, X_{ij}, of i and j together and dividing by X_i, the number of occurrences of i in the text. If the third word is similar to i or dog, P_{ik}/P_{jk} will be closer to 1. Else it will be closer to 0.

GloVe word embeddings allow us to define the distance between words based on their meaning, as can be seen in the visualization below. Words like 'dreams' are physically closer to words like 'memories' and 'visions' while unrelated words like 'glass' and 'milk' are located far away.

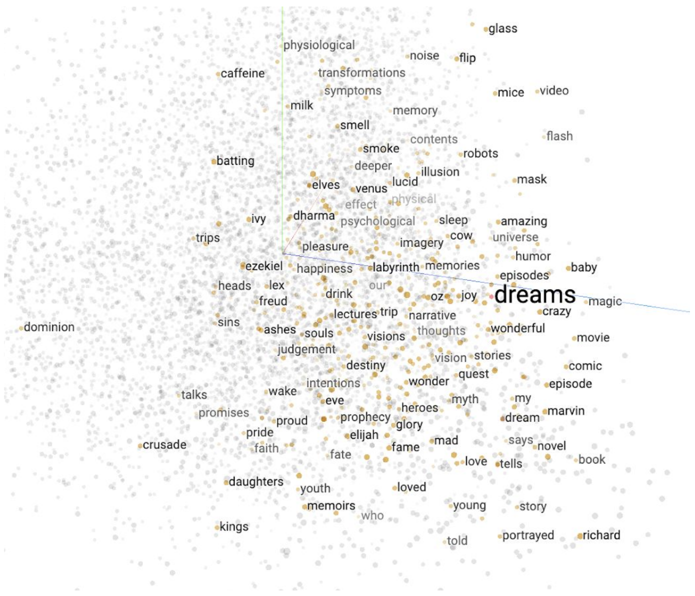


Figure 7. Visual representation of word embeddings

The overall system architecture for the textual sentiment analysis is as follows:

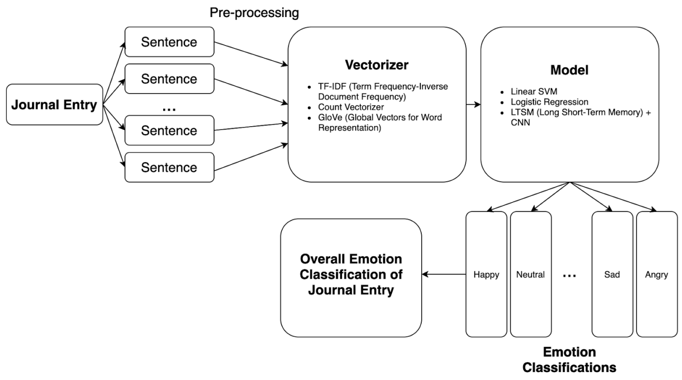

Pre-processing
Journal Entry
Sentence
Sentence
...
Sentence
Sentence
Vectorizer
TF-IDF (Term Frequency-Inverse Document Frequency)
Count Vectorizer
GloVe (Global Vectors for Word Representation)
Model
Linear SVM
Logistic Regression
LTSM (Long Short-Term Memory) + CNN
Overall Emotion Classification of Journal Entry
Happy
Neutral
...
Sad
Angry
Emotion Classifications

Figure 8. Block Diagram of Textual Sentiment Analysis System

After the user finishes saying their journal entry, the audio is isolated and split into sentences. This is done to achieve optimal performance when submitting the audio to the Google Speech Recognition library. While we could submit the entire journal entry audio, this library has the best transcription quality when the audio snippets are around 5 seconds each. Similarly, the LSTM network functions best when the input is a short sequence of text rather than one long journal entry. The most logical way of splitting the audio is by sentence, to maintain semantic meaning. However, Google Speech

Recognition does not have the ability to detect sentence boundaries. Rather the resulting transcription is just one unpunctuated block of text. To remedy this problem, we approximated the sentence boundaries by splitting the entry based on audio silence. Anytime the silence duration in the audio was above a chosen threshold, the audio was split.

After splitting the entry into audio clips, we transcribed each sentence using Google Speech Recognition. This is done through requests to Google's server, taking around 5-10 seconds each. To minimize the amount of time spent on transcription, we opted to launch threads for each sentence and parallelize this portion of the analysis.

Each sentence is then vectorized using the 200D GloVe word embeddings. After being vectorized it is fed into the LSTM-CNN (Long Short-Term Memory Convoluted Neural Network). The model applies one of the 6 emotion labels (or neutral emotion) to each sentence. The most frequently occuring emotion in the entry is then used as the overall emotion label

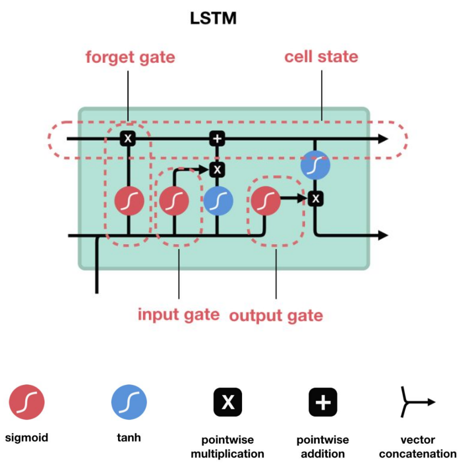


Figure 9. Layout of a basic LSTM cell (Moawad)

LSTMs and other recurrent neural networks work well with sequences of data like speech, audio and video as the models have knowledge of past inputs. LSTMs, however, benefit from having gates. As the model processes a sequence of the data, the output usually depends less and less on a very old input. LSTMs learn when to remember past inputs and when to forget. We chose to use a bidirectional LSTM. This model consists of two LSTMs, one reads inputs from past to future while the other reads from future to past. This works especially well for sequence-to-sequence data types like language. The significance of a word in a sentence depends on both the words that come before and after it.

Before any of this analysis can be done, the model must be trained. This requires a dataset and some data preprocessing. For the dataset, most programmers have two options: either a precompiled database or create our by scraping social media posts. Depending on how well the precompiled databases perform, we can decide on how to modify and combine different data to produce the optimal network. We chose to start off with a dataset consisting of 40,000 twitter messages, as social media tends to have emotionally charged content. We combined this with CrowdFlower's textual emotion database. One problem that we ran into was that the databases sometimes had an unbalanced number of samples across the different emotion categories. For example, neutral had almost 90% of entries in some databases. This meant we had to cut down the number of samples for other categories. Additionally, the data was not always highly accurate. Neutral sentences most of the time had some kind of emotion, so they were not truly neutral. Emotions like fear, anger and disgust were commonly confused. This is expected as different human emotions can be expressed with the same text.

Before we can start training, some preprocessing must be done to reduce noise in the data. This includes removing, upper casing, punctuation and stop words. These textual elements do not contribute much meaning to a sentence and only make it harder for a network to learn. This step is analogous to a gaussian blur in image recognition training. Some sources online also suggested removing the rarest words in an effort to decrease noise in the data.

Training first involves creating an embedding layer. We first compute an index mapping words to known embeddings using GloVe's precomputed vectors. If there are any new words, a randomized vector is created. The words are then fed into an embedding layer and then an LSTM layer. This LSTM layer is fed into the CNN, made up of 1 dimensional convolution layers. Previous research has shown that a CNN kernel size between 1 and 10 have had the best results. The data is then fed into a max pooling layer. The model is used to fit the data using a categorical cross-entropy loss function as our data is single-labeled, only one class can apply to each datapoint. For testing and validation, k-fold cross validation is not needed given the abundance of data. We can have separating training and testing data sets. When classifying textual emotion, the inputted data must go through the same if not similar preprocessing and vectorization to be fed into the classifier model.

Throughout the training process, the model struggled with identifying neutral emotions. Individually, the accuracies for identifying other emotions were above our 65% requirement. The neutral emotion is too subtle for the network to differentiate and the data used for this category was of low quality. If we had more time and better resources, we could have maybe improved the overall accuracy to be above 65%.

Table IV. Sentiment analysis emotion accuracies

Neutral	31%
Joy	71%
Anger	73%
Sadness	72%
Fear	70%
Disgust	70%
Surprised	71%

C. *Web Application*

The web application is built using the Django Python web-framework. The web application allows multiple users to create, record, and submit their journal entries for evaluation directly. The video recording through the web application is implemented through the use of the GetUserMedia Web API and is sent to the application's server using AJAX requests. The byte data of the user's recording is sent as chunks and the backend of the application reassembles the video, links it to the user using Django's model field, and saves it in storage. The web application and the emotion analysis models are both stored on the same server, allowing the web application to run the analysis nearly immediately after receiving the entry. When the user wants to view their old entries, the application retrieves the paths of the entries associated with the user and displays the list of videos to the user in the associated page. The emotion label and the creation date which are stored along with the video through the use of the model field are also sent with the list, allowing the user to see the associated labels and the time and date with the entries.

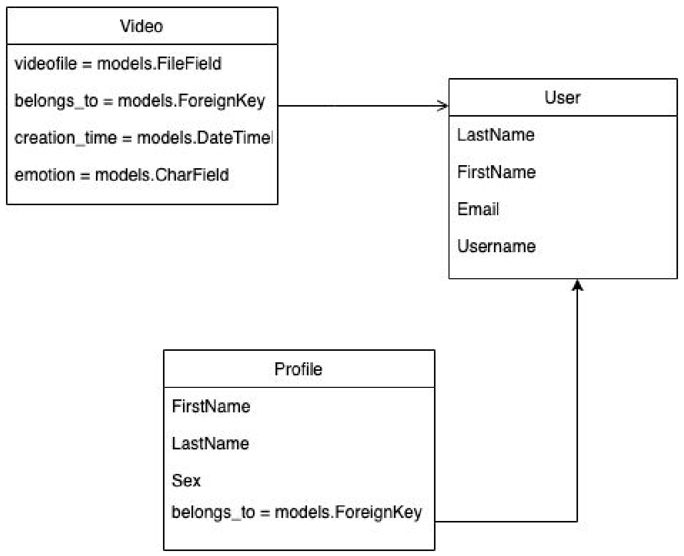

Video
videofile = models.FileField
belongs_to = models.ForeignKey
creation_time = models.DateTime
emotion = models.CharField
User
LastName
FirstName
Email
Username
Profile
FirstName
LastName
Sex
belongs_to = models.ForeignKey

Figure 10. Models used in application

D. *Voice Tone Analysis*

An additional component to our project to aid emotion analysis is voice tone recognition. Complementing sentiment analysis, voice tone recognition takes the audio recording and finds features from the sound waves, notably the pitch. Similar to the way muscles can tense up or relax based on emotion, vocal folds, which are in charge of creating the various pitches and tones humans make, do the same. The tightening or loosening of vocal folds changes the frequency of the vibration. The frequency of the vibration of the vocal folds determines pitch. Pitch is used to calculate the Mel Frequency Cepstrum Coefficients. The mel frequency cepstrum approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum, and is therefore used as features in many speech recognition systems. We calculate the MFCC coefficients on the short clippings using Python's Librosa library.

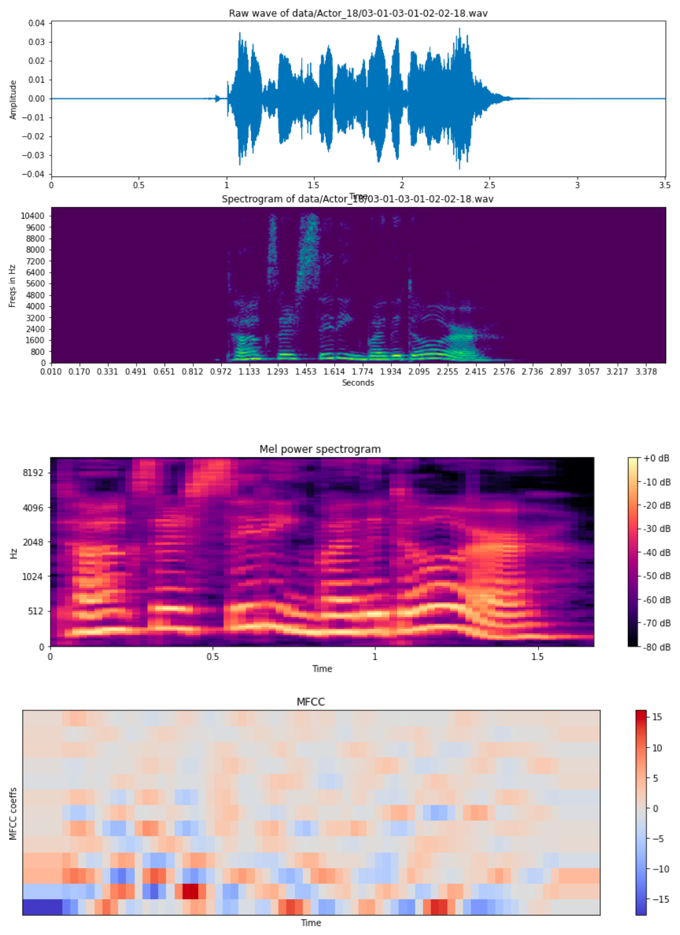
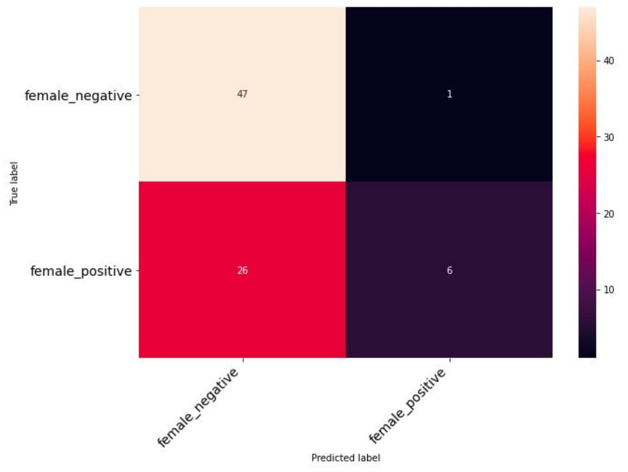


Figure 11. Graphs of voice recordings, associated spectrograph, Mel power spectrograph, and MFCC coefficients

To train our network, we used the RAVDESS Audio-Visual Database of Emotional Speech dataset, composed of 1440 files from 24 professional voice actors depicting different emotions. The features used to train were the MFCC coefficients. Our neural network is made up of 1D convolution layers, max pooling layers, and batch normalization layers. Some factors had to be considered when training, such as the sex of the speaker. Because male and female voices have a large difference in pitch, we had to split our dataset into male and female and train separately. As such, we have two algorithms for voice tone analysis, and we ask the user for their sex when they make an account on the web application. Though we initially wanted to be able to predict from seven emotions, the accuracies were too low and did not meet our requirements. We tried different approaches such as different input sizes and reducing the number of emotions, but at the end we took inspiration from the approach taken by article[2] and chose to use binary analysis, i.e. classifying emotions into positive or negative. The male binary analyzer had a 60% test accuracy and 66.25% accuracy for female voices.



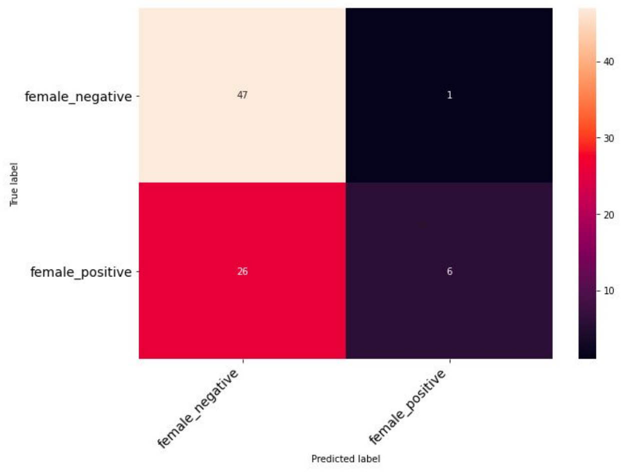


Figure 12. Performances for male and female binary classifiers

VII. Project Management

A. Schedule

Appendix I contains our schedule, separated into different categories:

- ❏ Facial emotion recognition
- ❏ Sentiment analysis
- ❏ Tone recognition
- ❏ Web application

The timeline has not changed from the original proposed schedule, but the roles assigned have changed. Specifically, the task of training has changed from all team members to one person per type of analysis to increase efficiency.

B. Team Member Responsibilities

Patrick Gao primarily focused on the facial emotion analysis portion of the project. Vinay Kukutla focused on text sentiment analysis, and Yoojin Lee on tone recognition and the web application. The entire team worked together to integrate the different components for the final product.

C. Budget

Although materials were purchased for our original plan to build a physical prototype, we did not end up using these materials for our final product following our refocus steps. Expenses were also used for AWS credits but are not outlined here. Refer to section IX for the breakdown of AWS credit usage.

Table IV. Budget breakdown.

Item	Price
Raspberry Pi	$45
Camera	$22
Microphone	$10

D. Refocus

The COVID-19 outbreak was a surprise to all. With the transition to online classes, it was necessary to restructure our project. The most significant change was removing the physical portion of EMO. We would no longer run the client code on a Raspberry PI with a custom enclosure. Instead, we chose to run the code on a laptop, utilizing its camera and mic. This also meant on longer incorporating IoT device management into EMO. Our web application became the primary user interface on which users were to start and submit recordings.

VIII. Related Work

E-MO is similar to products such as Google Home and Amazon Alexa, but those products respond to direct vocal commands only. E-MO is different for a couple of reasons; E-MO acts without verbal commands, taking cues from your emotion instead. Additionally, Google Home, Alexa and most other smart home products are blind to the user's emotions. E-MO utilizes an user's emotional state to provide them a suitable environment.

IX. AWS Credit Usage

$6.36 of our credits have been charged so far as of 05/01/20 for creating a test instance for the web app. $22.54 of our AWS credits are estimated for the month of May for on-demand and spot-request m4.large instances and EBS storage for our web application, user journal storage, and to host our networks on the cloud.

Thank you to CMU and Amazon for making this project possible.

IX. References

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau. Sentiment Analysis of Twitter Data.

[2] Chu, Reza. "Speech Emotion Recognition with Convolutional Neural Network." and associated GitHub repository. Medium, Towards Data Science, 1 June 2019

[3] Dasgupta, Poorna. "Detection and Analysis of Human Emotions through Voice and Speech Pattern Processing." DeepAI, International Journal of Computer Trends and Technology (IJCTT), 27 Oct. 2017,

[4] J. Pennington, R. Socher, C Manning. GloVe: Global Vectors for Word Representation.

[5] K. Simonyan and A. Zisserman. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[6] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLOS ONE 13(5): e0196391.

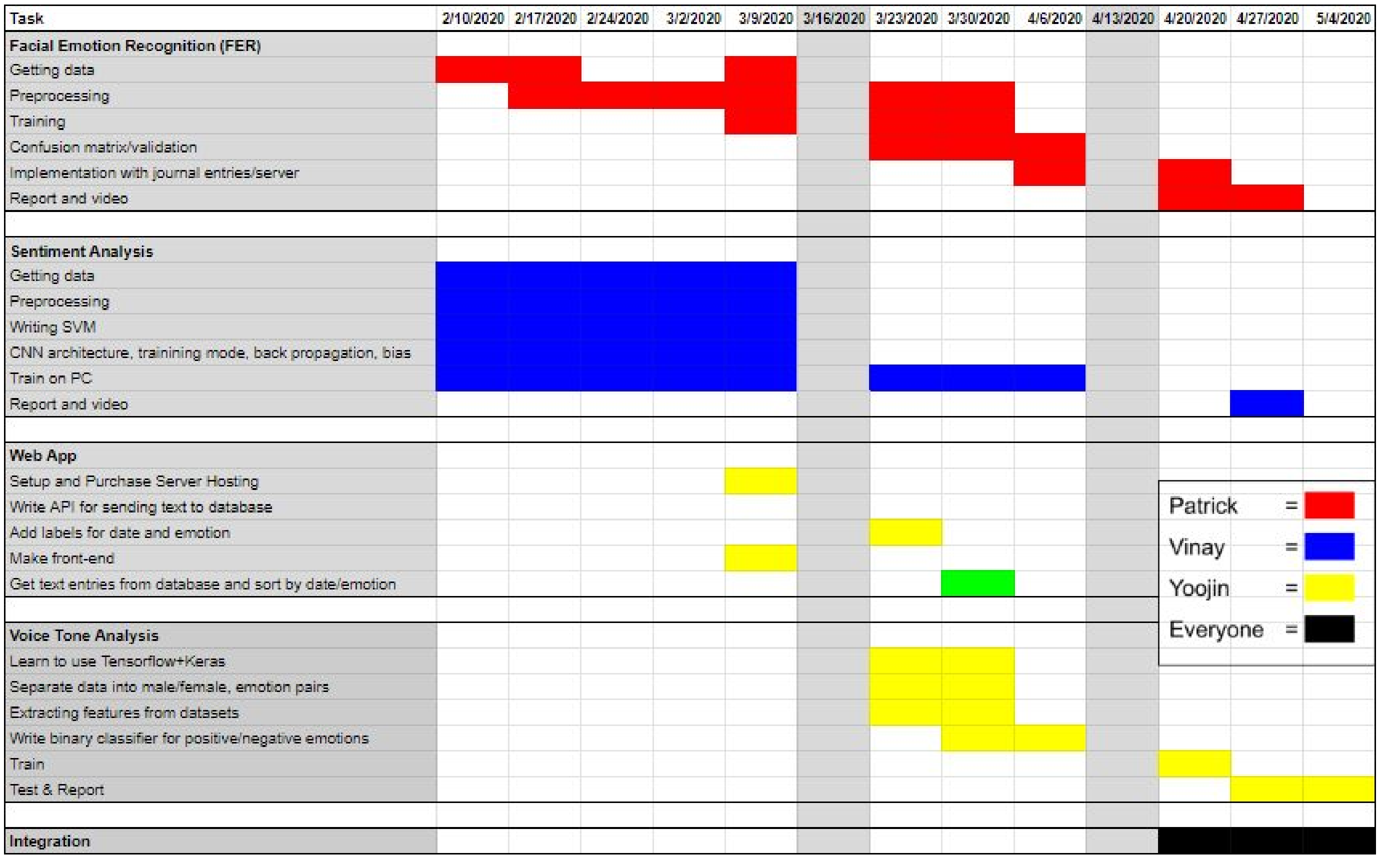
[7] Mollahosseini, Ali, Behzad Hasani, and Mohammad H. Mahoor. "AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild." IEEE Transactions on Affective Computing 10.1 (2019): 18–31. Crossref. Web.

[8] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, San Francisco, CA, 2010, pp. 94-101.

[9] Ravi, Aravind. (2018). Pre-Trained Convolutional Neural Network Features for Facial Expression Recognition.

[10] S. Guzman. Determinants of Politically Charged News in Media: Relationships Between Entity and Sentiment Tonality.

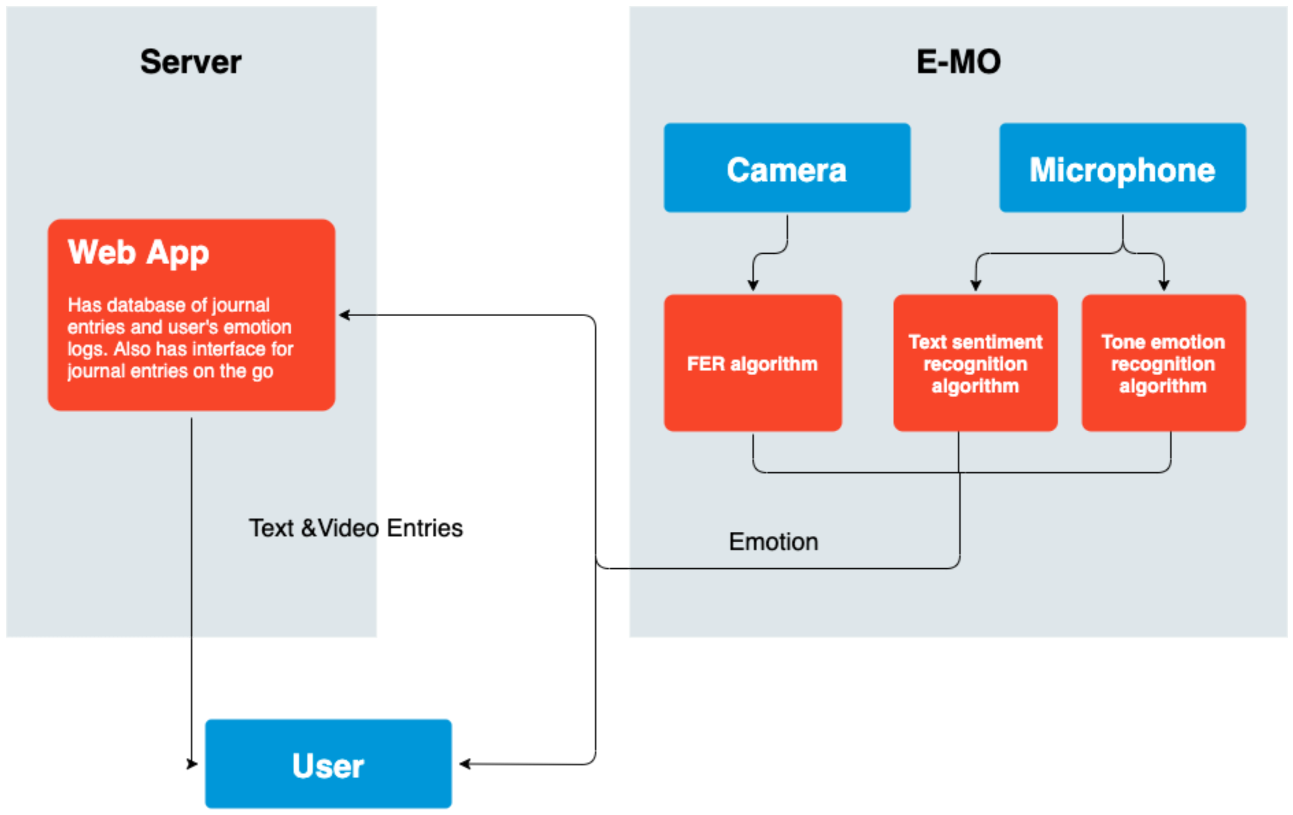
X. Appendix I. Schedule



Task	2/10/2020	2/17/2020	2/24/2020	3/2/2020	3/9/2020	3/16/2020	3/23/2020	3/30/2020	4/6/2020	4/13/2020	4/20/2020	4/27/2020	5/4/2020
Facial Emotion Recognition (FER)													
Getting data	🟥	🟥			🟥								
Preprocessing		🟥	🟥	🟥	🟥		🟥	🟥					
Training					🟥		🟥	🟥					
Confusion matrix/validation							🟥	🟥	🟥				
Implementation with journal entries/server									🟥		🟥		
Report and video											🟥	🟥	
Sentiment Analysis													
Getting data	🟦	🟦	🟦	🟦	🟦								
Preprocessing	🟦	🟦	🟦	🟦	🟦								
Writing SVM	🟦	🟦	🟦	🟦	🟦								
CNN architecture, trainining mode, back propagation, bias	🟦	🟦	🟦	🟦	🟦								
Train on PC	🟦	🟦	🟦	🟦	🟦		🟦	🟦	🟦				
Report and video												🟦	
Web App													
Setup and Purchase Server Hosting					🟨								
Write API for sending text to database													
Add labels for date and emotion							🟨						
Make front-end					🟨								
Get text entries from database and sort by date/emotion								🟩					
Voice Tone Analysis													
Learn to use Tensorflow+Keras							🟨	🟨					
Separate data into male/female, emotion pairs							🟨	🟨					
Extracting features from datasets							🟨	🟨					
Write binary classifier for positive/negative emotions								🟨	🟨				
Train											🟨		
Test & Report												🟨	🟨
Integration											⬛	⬛	⬛

Patrick	=	🟥
Vinay	=	🟦
Yoojin	=	🟨
Everyone	=	⬛

XI. Appendix II. High Level System Block Diagram


Server
Web App
Has database of journal entries and user's emotion logs. Also has interface for journal entries on the go
Text &Video Entries
E-MO
Camera
Microphone
FER algorithm
Text sentiment recognition algorithm
Tone emotion recognition algorithm
Emotion
User