ESAPR

Emotional State Analyser for Psychological Response

Project Report

Submitted by

(Roll No: **12619001010**, Reg No: 029238 of 2019-2020)
(Roll No: **12619001087**, Reg No: 031543 of 2019-2020)
(Roll No: **12619001036**, Reg No: 013597 of 2019-2020)

Under the Supervision of

Prof. Mohuya Byabartta Kar

Department Of Computer Science And Engineering

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

HERITAGE INSTITUTE OF TECHNOLOGY

KOLKATA

BONAFIDE CERTIFICATE

Certified that this Project "**EMOTIONAL STATE ANALYSER FOR PSYCHOLOGICAL RESPONSE (ESPAR)**" is the bonafide work of "**Aditya Roshan Jha**", "**Anushka Das**" and "**Moonmoon Mondal**". Who carried out project work under my supervision.

SIGNATURE

HEAD OF THE DEPARTMENT

Computer Sci. & Enggi.

Heritage Institute Of Technology

SIGNATURE

PROJECT GUIDE

Computer Sci. & Enggi.

Heritage Institute Of Technology

SIGNATURE EXAMINER(S)

Acknowledgement

The authors remember with gratitude the constant guidance, suggestions and encouragement forwarded by several respected persons. We would like to convey my heartfelt gratitude to **Prof. Dr. Mohuya Byabartta Kar** for her tremendous support and assistance in the completion of my project. It was in the 7th semester final project report submission and evaluations that we received valuable feedback from **Prof. Dr. Sabhyasachi Banerjee.** The completion of the project would not have been possible without their help and insights.

The authors would like to convey their gratitude to all the professors in the Bachelor Of Technology course in the Computer Science and Engineering department of Heritage Institute Of Technology for excellent teaching that helped to partially complete the project smoothly.

Finally the authors wishes to express a heartfelt gratitude to our team members, who worked tirelessly to complete the project in the time bound manner with the great hard work and ingenuity.

Aditya Roshan Jha

Anushka Das

Moonmoon Mondal

ABSTRACT

This project have the ultimate main to reduce psychological burden that people of these time face. At the end of the project we would have a working application platform which would be available as subsytem in the form of Web and Android (if not iOS). This is an emotion processing application which is going to identify the emotion of the user and will therefore might help to recommend list of songs, web-articles as well as videos on the basis of it. The objective is to design an efficient machine learning application by using the images of the user's facial expressions. This application first takes images of the user's face and the text messages written by the user and then applies sentiment analysis to generate the overall reaction of the user using the platform that how the assess the performance on the sentimental basis.We are exploring state of the art models in multimodal emotion recognition. We have chosen to explore textual, sound and video inputs and develop an ensemble model that gathers the information from all these sources and displays it in a clear and interpretable way.

We are naming this project in acronym of **ESAPR** (Emotional State Analyser for Pschyological Response).

Index

This page is

intentionally

left blank.

Introduction

Recognizing facial expression of human is of significance in certain domains. Based on the expression of humans we can predict the type of content which affects a human and gain insights about their face expressions when exposed to a certain content. The human facial expression consists of seven states which are angry, disgust, fear, happy, neutral, sad and surprise. Each of these facial expressions have unique features and are important in classifying them. Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service, or idea. It involves the use of data mining, machine learning (ML) and artificial intelligence (AI) to mine text for sentiment and subjective information.

Multimodal Emotion Recognition Multimodal Emotion Recognition is a relatively new discipline that aims to include text inputs, as well as sound and video. This field has been rising with the development of social networks that gave researchers access to a vast amount of data. Recent studies have been exploring potential metrics to measure the coherence between emotions from the different channels.

Requirement Specifications

Hardware

- ESAPR should capture video from the camera source, that is front camera of Mobile Device and the webcam of the PC and display the resulting findings in the form of videos, music and web-articles.
- Graphic Card of at least 8GB, RAM of at least 16GB, at least Ryzen 5 3500U or Intel i5 9th Gen required for the overall system to run on the local machine. As if deployed this requirement would be eliminated altogether.
- It would be great if the it is used as micro-service while being deployed on the cloud based server.

Software

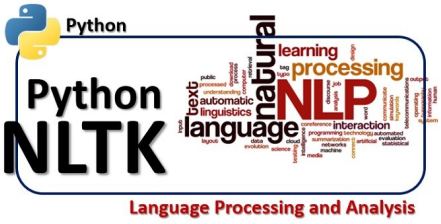
- ESAPR should be able to perform all this in the realtime. Even though the computing power of the different devices might not be good enough.
- ESAPR will use internet and cloud based approach to achieve the required results, at the same time not possibly creating hindrance to other software systems in the device.
- Some of the libraries and framework used might be legacy system, for the beta version of it, support for the legacy versions is required.

Tools And Frameworks used till now










Problem Definition

We are trying to provide definitions of affective computing and multimodal sentiment analysis in the context of our project. Those definitions may vary depending on the context.

Affective Computing Affective computing is a field of Machine Learning and Computer Science that studies the recognition and the processing of human affects.

Multimodal Emotion Recognition Multimodal Emotion Recognition is a relatively new discipline that aims to include text inputs, as well as sound and video. This field has been rising with the development of social networks that gave researchers access to a vast amount of data. Recent studies have been exploring potential metrics to measure the coherence between emotions from the different channels. We are going to explore several categorical targets depending on the input considered.

We are going to explore several categorical targets depending on the input considered.

Data Type	**Categorical Target**
Textual	Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism
Sound	Happy, Sad, Angry, Fearful, Surprise, Neutral and Disgust
Video	Happy, Sad, Angry, Fearful, Surprise, Neutral and Disgust

For the text inputs, we are going to focus on the so-called Big Five, widely used in personality surveys.

Methodology

Our aim is to develop a model able to provide real time sentiment analysis with a visual user interface using Tensorow.js technology.

Therefore, we have decided to separate two types of inputs :

1. Textual input, such as answers to questions that would be asked to a person from the platform

2. Video input from a live webcam or stored from an MP4 or WAV file, from which we split the audio and the images.

Data Sources

We have chosen to diversify the data sources we used depending on the type of data considered. All data sets used are free of charge and can be directly downloaded.

- For the text input, we are using the Stream-of-consciousness dataset that was gathered in a study by Pennebaker and King [1999]. It consists of a total of 2,468 daily writing submissions from 34 psychology students (29 women and 5 men whose ages ranged from 18 to 67 with a mean of 26.4). The writing submissions were in the form of a course unrated assignment. For each assignment, students were expected to write a minimum of 20 minutes per day about a specific topic. The data was collected during a 2-week summer course between 1993 to 1996. Each student completed their daily writing for 10 consecutive days. Students' personality scores were assessed by answering the Big Five Inventory (BFI) [John et al., 1991]. The BFI is a 44-item self-report questionnaire that provides a score for each of the five personality traits. Each item consists of short phrases and is rated using a 5-point scale that ranges from 1 (disagree strongly) to 5 (agree strongly). An instance in the data source consists of an ID, the actual essay, and five classification labels of the Big Five personality traits. Labels were originally in the form of either yes ('y') or no ('n') to indicate scoring high or low for a given trait.

- For audio data sets, we are using the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). This database contains 7356 files (total size: 24.8 GB). The database contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity(normal, strong), with an additional neutral expression. All conditions are avail-able in three modality formats: Audio-only (16bit, 48kHz .wav), Audio-Video(720p H.264, AAC 48kHz, .mp4), and Video-only (no sound)." https://zenodo.org/record/1188976#.XCx-tc9KhQI

- For the video data sets, we are using the popular FER2013 Kaggle Challenge data set. The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The data set remains quite challenging to use, since there are empty pictures, or wrongly classified images. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

Speech Emotion Recognition

The aim of this section is to explore speech emotion recognition techniques from an audio recording or live speech. Although in the website of our project you will only have the option for going all out live.

Data

The data set used for training is the Ryerson Audio-Visual Database of Emotional Speech and Song: https://zenodo.org/record/1188976#.XA48aC17Q1J

RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

RAVDESS								
Emotions	Happy	Sad	Angry	Scared	Dis-gusted	Sur-prised	Neutral	**Total**
Man	*96*	*96*	*96*	*96*	*96*	*96*	*96*	**672**
Woman	*96*	*96*	*96*	*96*	*96*	*96*	*96*	**672**
Total	*192*	*192*	*192*	*192*	*192*	*192*	*192*	**1344**

Requirements

```
Python : 3.6.5
Scipy : 1.1.0
Scikit-learn : 0.20.1
Tensorflow : 1.12.0
Keras : 2.2.4
Numpy : 1.15.4
Librosa : 0.6.3
Pyaudio : 0.2.11
Ffmpeg : 4.0.2
```

Notebooks

Notebooks provided for this subsystem.

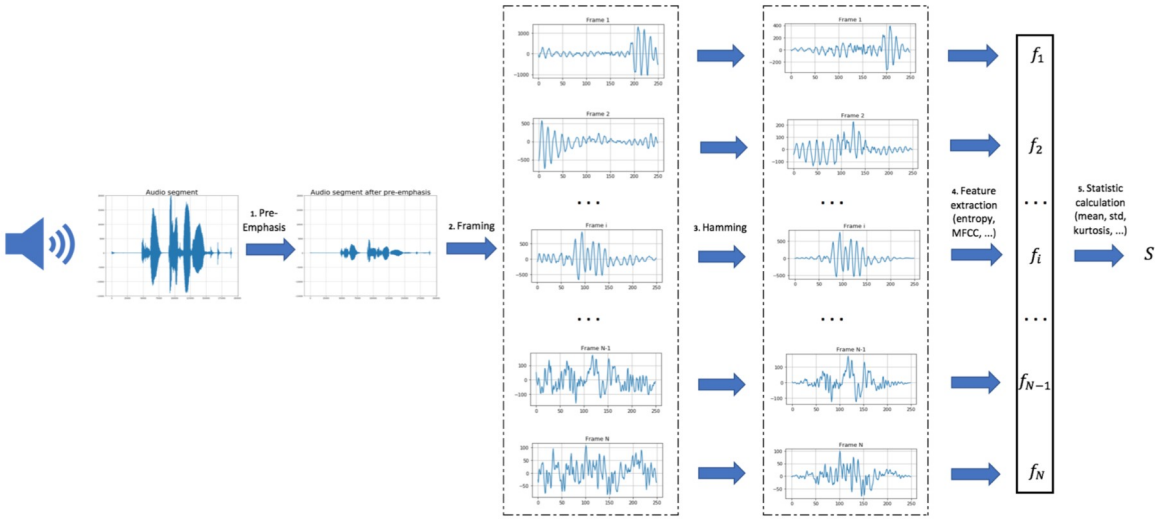
- `01 - Preprocessing[SVM].ipynb` : Signal preprocessing and feature extraction from time and frequency domain (global statistics) to train SVM classifier.
- `02 - Train [SVM].ipynb` : Implementation and training of SVM classifier for Speech Emotion Recognition

- `01 - Preprocessing[CNN-LSTM].ipynb` : Signal preprocessing and log-mel-spectrogram extraction to train TimeDistributed CNNs
- `02 - Train [CNN-LSTM].ipynb` : Implementation and training of TimeDistributed CNNs classifier for Speech Emotion Recognition

Models

SVM

Classical approach for Speech Emotion Recognition consists in applying a series of filters on the audio signal and partitioning it into several windows (fixed size and time-step). Then, features from time domain (Zero Crossing Rate, Energy and Entropy of Energy) and frequency domain (Spectral entropy, centroid, spread, flux, rolloff and MFCCs) are extracted for each frame. We compute then the first derivatives of each of those features to capture frame to frame changes in the signal. Finally, we calculate the following global statistics on these features: *mean, median, standard deviation, kurtosis, skewness, 1% percentile, 99% percentile, min, max* and *range* and train a simple SVM classifier with rbf kernel to predict the emotion detected in the voice.



SVM classification pipeline:

1.Voice recording

2.Audio signal discretization

3.Apply pre-emphasis filter

4.Framing using a rolling window

5.Apply Hamming filter

6.Feature extraction

7.Compute global statistics

8.Make a prediction using our pre-trained model

01 – Preprocessing[SVM] : Signal Preprocessing

Context

The aim of this notebook is to set up all speech emotion recognition preprocessing and audio features extraction.

Audio features:

The complete list of the implemented short-term features is presented below:

- **Zero Crossing Rate**: The rate of sign-changes of the signal during the duration of a particular frame.
- **Energy**: The sum of squares of the signal values, normalized by the respective frame length.
- **Entropy of Energy**: The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
- **Spectral Centroid**: The center of gravity of the spectrum.
- **Sprectral Spread**: The second central moment of the spectrum.
- **Spectral Entropy**: Entropy of the normalized spectral energies for a set of sub-frames.
- **Spectral Flux**: The squared difference between the normalized magnitudes of the spectra of the two successive frames.
- **Spectral Rolloff**: The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
- **MFCCS**: Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.

Global Statistics are then computed on upper features:

- **mean, std, med, kurt, skew, q1, q99, min, max and range**

02 - Train [SVM].ipynb : SVM Classifier

Context

The aim of this notebook is to set up all speech emotion recognition preprocessing and audio features extraction.

Audio features:

The complete list of the implemented short-term features is presented below:

- **Zero Crossing Rate**: The rate of sign-changes of the signal during the duration of a particular frame.
- **Energy**: The sum of squares of the signal values, normalized by the respective frame length.
- **Entropy of Energy**: The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
- **Spectral Centroid**: The center of gravity of the spectrum.
- **Sprectral Spread**: The second central moment of the spectrum.
- **Spectral Entropy**: Entropy of the normalized spectral energies for a set of sub-frames.
- **Spectral Flux**: The squared difference between the normalized magnitudes of the spectra of the two successive frames.
- **Spectral Rolloff**: The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
- **MFCCS**: Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.

Global Statistics are then computed on upper features:

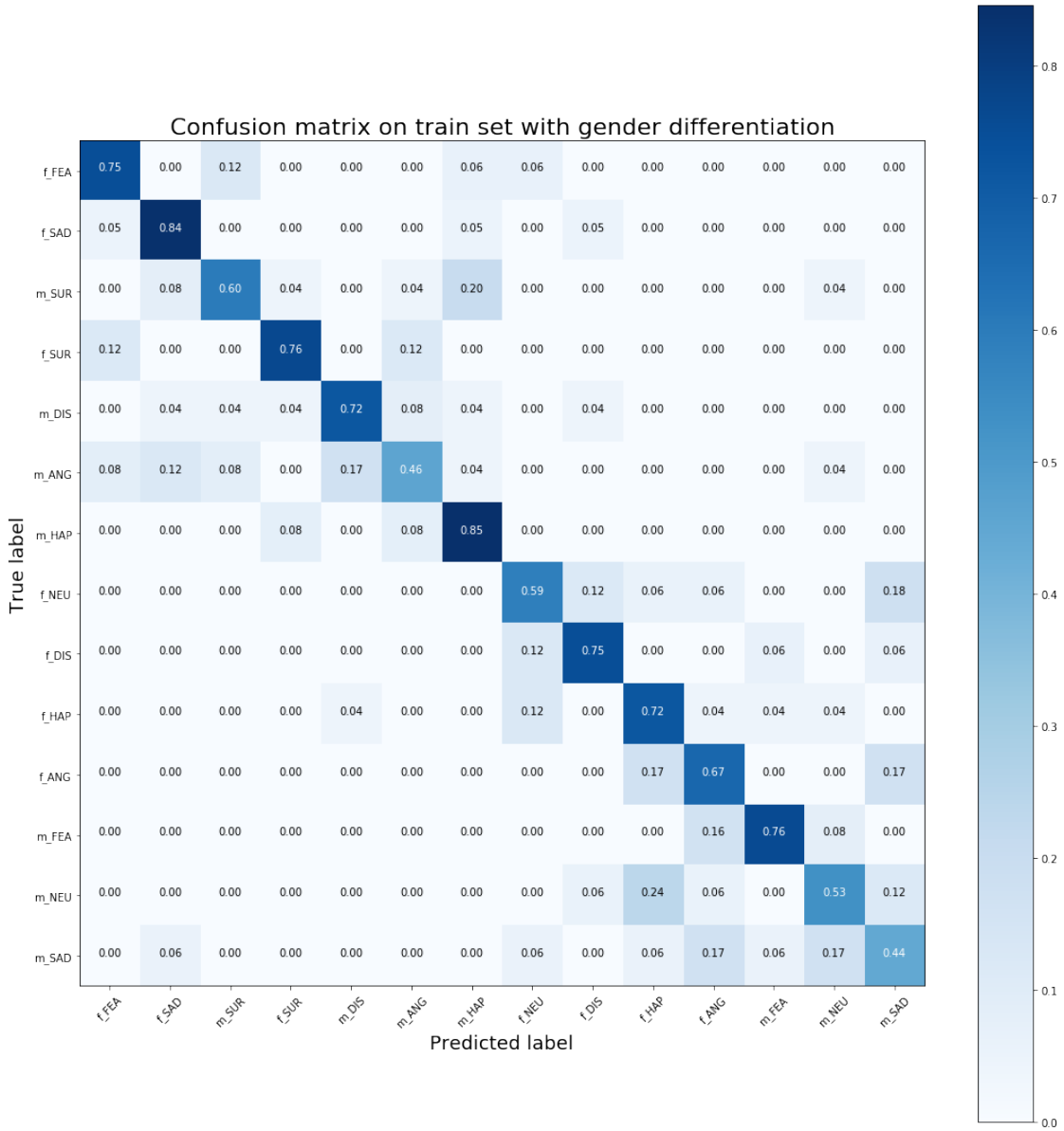
- **mean, std, med, kurt, skew, q1, q99, min, max and range**

VIII. Best model prediction

```
# Confusion matrix plot function
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=22)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
```

```
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label', fontsize=18)
    plt.xlabel('Predicted label', fontsize=18)
    plt.tight_layout()

# Fit best mode
model = SVC(kernel='rbf', C=3, gamma=0.005, decision_function_shape='ovr').fit(X_train, y_train)

# Prediction
pred = model.predict(X_test)

# Score
score = model.score(X_test, y_test)

# Reverse label encoder
pred = (lb.inverse_transform((pred.astype(int).flatten())))
actual = (lb.inverse_transform((y_test.astype(int).flatten())))

# Build dataFrame
df_pred = pd.DataFrame({'Actual': actual, 'Prediction': pred})

# Print Score
print('Accuracy Score on test dataset: {}%'.format(np.round(100 * score,2)))

# Compute confusion matrix
confusion = confusion_matrix(actual, pred)

# Plot non-normalized confusion matrix
plt.figure(figsize=(15, 15))
plot_confusion_matrix(confusion, classes=set(actual),normalize=True,
                      title='Confusion matrix on train set with gender differentiation')
```

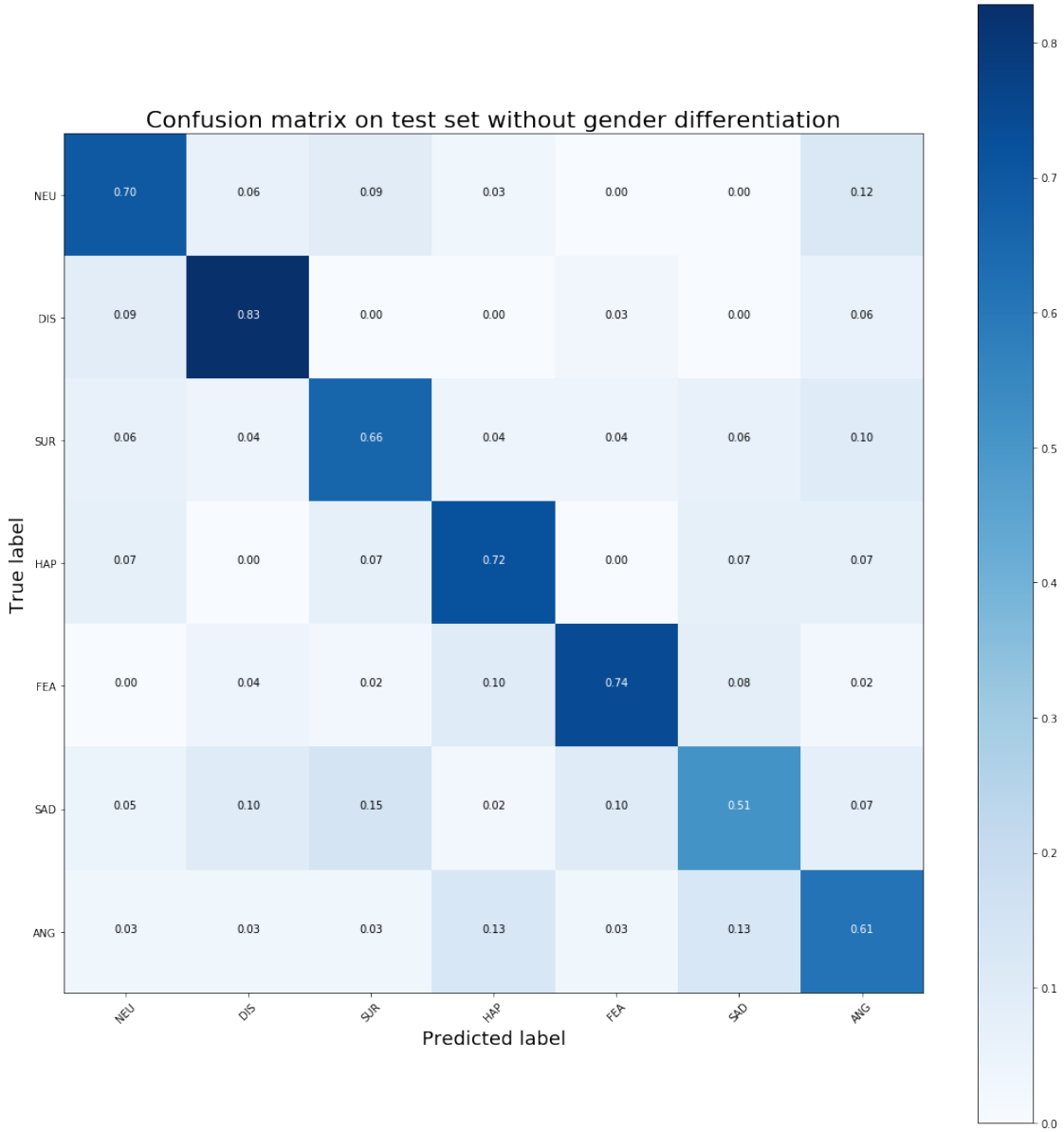
```
Accuracy Score on test dataset: 66.91%
```




```
# Compute prediction without gender differentation
PRED = list(map(lambda i:i[2:], pred))
ACTUAL = list(map(lambda i:i[2:], actual))

# Compute related prediction score
SCORE = accuracy_score(ACTUAL, PRED)

# Print Score
print('Accuracy Score on test dataset: {}%'.format(np.round(100 * SCORE,2)))

# Compute confusion matrix
confusion = confusion_matrix(ACTUAL, PRED)
```

```
# Plot non-normalized confusion matrix
plt.figure(figsize=(15, 15))
plot_confusion_matrix(confusion, classes=set(ACTUAL),normalize=True,
                      title='Confusion matrix on test set without gender differentiation')
```

Accuracy Score on test dataset: 68.03%



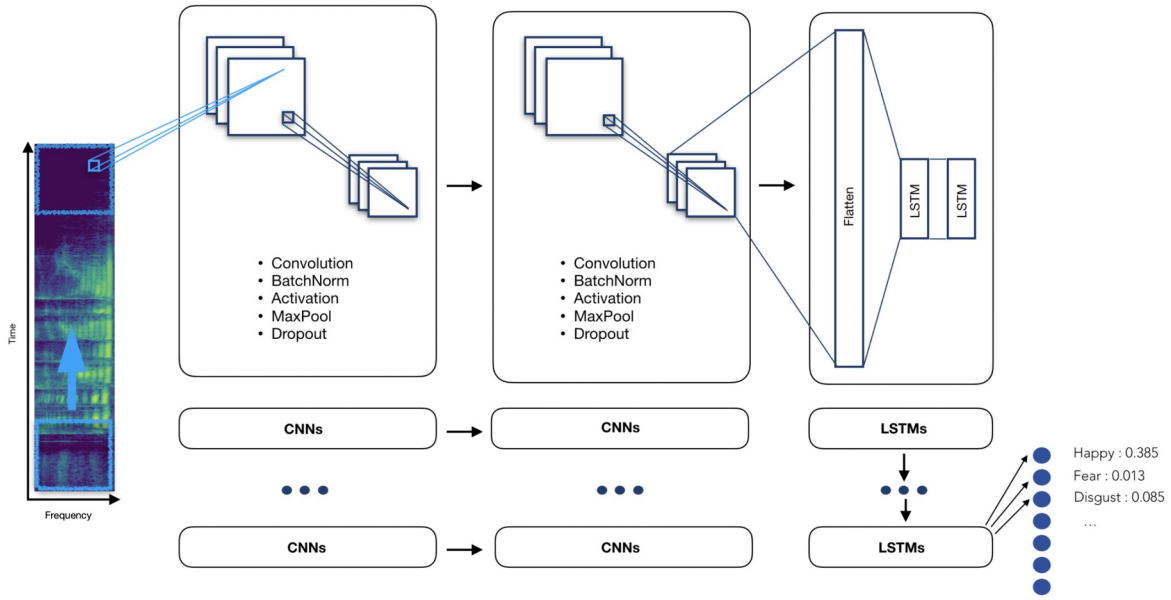
IX. Save model

```
# save the model to local
pickle.dump(model, open('../Model/MODEL_CLASSIFIER.p', 'wb'))
```

```
# Save label encoder
pickle.dump(lb, open("../Model/MODEL_ENCODER.p", "wb"))

# Save PCA
pickle.dump(pca, open("../Model/MODEL_PCA.p", "wb"))

# Save MEAN and STD of each features
MEAN = features.mean(axis=0)
STD = features.std(axis=0)
pickle.dump([MEAN, STD], open("../Model/MODEL_SCALER.p", "wb"))

# Save feature parameters
stats = ['mean', 'std', 'kurt', 'skew', 'q1', 'q99']
features_list = ['zcr', 'energy', 'energy_entropy', 'spectral_centroid', 'spectral_spread',
'spectral_entropy', 'spectral_flux', 'sprectral_rolloff']
win_step = 0.01
win_size = 0.025
nb_mfcc = 12
diff = 0
PCA = True
DICO = {'stats':stats, 'features_list':features_list, 'win_size':win_size, 'win_step':win_step,
'nb_mfcc':nb_mfcc, 'diff':diff, 'PCA':PCA}
pickle.dump(DICO, open("../Model/MODEL_PARAM.p", "wb"))
```

TimeDistributed CNNs

The main idea of a Time Distributed Convolutional Neural Network is to apply a rolling window (fixed size and time-step) all along the log-mel-spectrogram. Each of these windows will be the entry of a convolutional neural network, composed by four Local Feature Learning Blocks (LFLBs) and the output of each of these convolutional networks will be fed into a recurrent neural network composed by 2 cells LSTM (Long Short Term Memory) to learn the long-term contextual dependencies. Finally, a fully connected layer with *softmax* activation is used to predict the emotion detected in the voice.



TimeDistributed CNNs pipeline:

1. Voice recording
2. Audio signal discretization
3. Log-mel-spectrogram extraction
4. Split spectrogram with a rolling window
5. Make a prediction using our pre-trained model

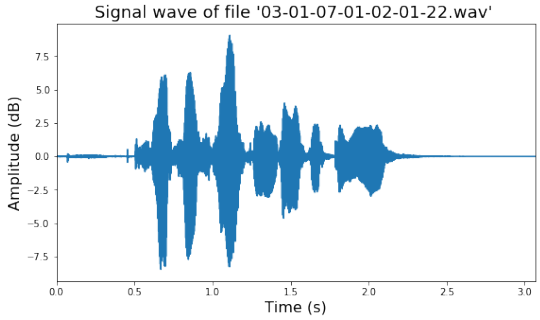
01 – Preprocessing[CNN-LSTM].ipynb : Signal Preprocessing

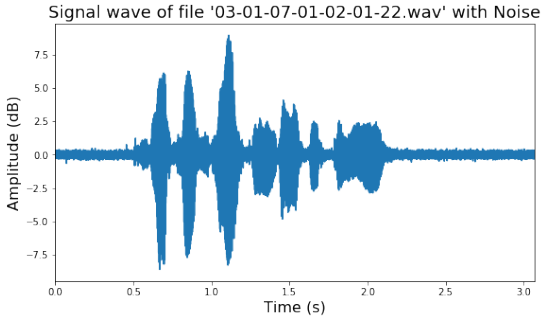
Context

The aim of this notebook is to set up all speech emotion recognition preprocessing for the time distributed ConvNet.

The signal preprocessing include :

- Signal discretization
- Audio data augmentation
- Log-mel-spectrogram extraction
- Time distributed framing
- Build train and test data set



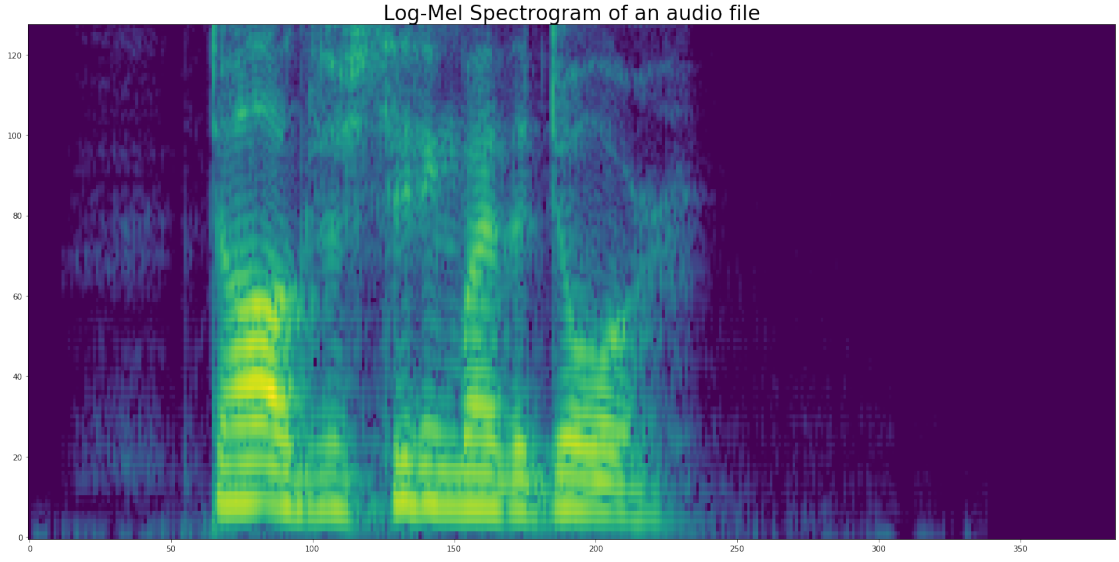



```
Audio file '03-01-07-01-02-01-22.wav':
```

```
<IPython.lib.display.Audio object>
```

```
Audio file '03-01-07-01-02-01-22.wav' with noise:
```

```
<IPython.lib.display.Audio object>
```



Save as

```
# Save Train and test set
pickle.dump(X_train.astype(np.float16), open('../Datas/Pickle/RAVDESS/DIS/[RAVDESS][MEL_SPECT][X_train].p', 'wb'))
pickle.dump(y_train, open('../Datas/Pickle/RAVDESS/DIS/[RAVDESS][MEL_SPECT][y_train].p', 'wb'))
pickle.dump(X_test.astype(np.float16), open('../Datas/Pickle/RAVDESS/DIS/[RAVDESS][MEL_SPECT][X_test].p', 'wb'))
pickle.dump(y_test, open('../Datas/Pickle/RAVDESS/DIS/[RAVDESS][MEL_SPECT][y_test].p', 'wb'))
```

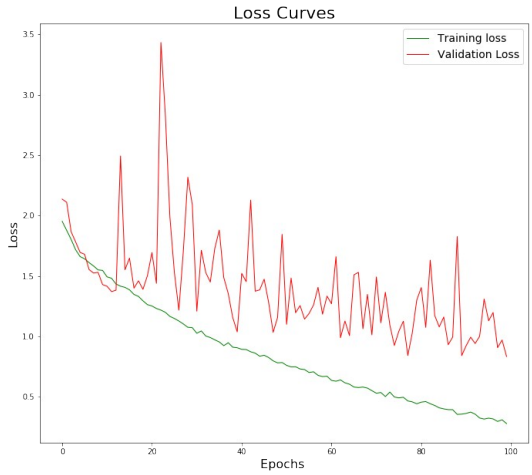
02 - Train [CNN-LSTM].ipynb : Time Distributed ConvNet

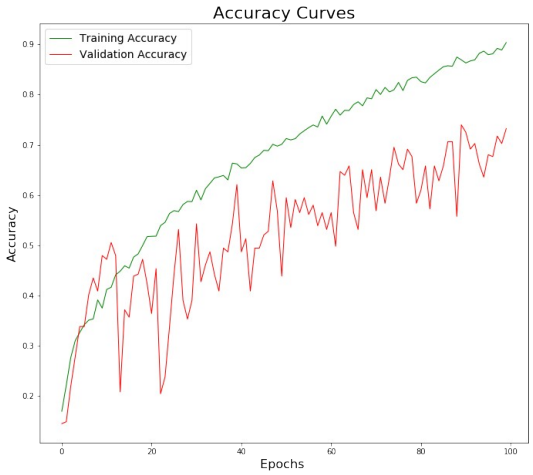
Context

In this project I build a prevision model using deep learning combining **CNN** and **LSTM** to detect a person's emotions (HAPPY, SAD, FEAR, ANGRY, DISGUST, SURPRISE, NEUTRAL) just by their voice.

Audio features:

- **Log-mel-spectrogram**: the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency





VII. Save model

```
model.save('drive/My Drive/SpeechEmotionRecognition/[CNN-LSTM]M.h5')
model.save_weights('drive/My Drive/SpeechEmotionRecognition/[CNN-LSTM]W.h5')
```

Performance

To limit overfitting during training phase, we split our data set into train (80%) and test set (20%). Following show results obtained on test set:

Model	Accuracy
SVM on global statistic features	**68.3%**
Time distributed CNNs	**76.6%**

Text-based Personality Traits Recognition

In this section you will find all resources, models and Python scripts relative to text-based personality traits recognition.

Emotion recognition through text is a challenging task that goes beyond conventional sentiment analysis : instead of simply detecting neutral, positive or negative feelings from text, the goal is to identify a set of emotions characterized by a higher granularity. For instance, feelings like anger or happiness could be included in the classification. As recognizing such emotions can turn out to be complex even for the human eye, machine learning algorithms are likely to obtain mixed performances. It is important to note that nowadays, emotion recognition from facial expression tends to perform better than from textual expression. Indeed, many subtleties should be taken into account in order to perform an accurate detection of human emotions through text, context-dependency being one of the most crucial. This is the reason why using advanced natural language processing is required to obtain the best performance possible.

In the context of our study, we chose to use text mining in order not to detect regular emotions such as disgust or surprise, but to recognize personality traits based on the "Big Five" model in psychology. Even though emotion recognition and personality traits classification are two separate fields of studies based on different theoretical underpinnings, they use similar learning-based methods and literature from both areas can be interesting. The main motivation behind this choice is to offer a broader assessment to the user : as emotions can only be understood in the light of a person's own characteristics, we thought that analyzing personality traits would provide a new key to understanding emotional fluctuations. Our final goal is to enrich the user experience and improve the quality of our analysis : any appropriate and complementary information deepening our understanding of the user's idiosyncrasies is welcome

Our main goal is to leverage on the use of statistical learning methods in order to build a tool capable of recognizing the personality traits of an individual given a text containing his answers to pre-established personal questions. Our first idea was to record a user's interview and convert the file from audio to text : in this way we would have been able to work with similar data for text, audio and video. Nevertheless, the good transcription of audio files to text requires the use of expensive APIs, and the tools available for free in the market don't provide sufficient quality. This is the reason why we chose to apply our personality traits detection model to short texts directly written by users : in this way we can easily target particular themes or questions and provide indications of the language level to use. As a result of this, we can make sure that the text data we use to perform the personality traits detection is consistent with the data used for training, and therefore ensure the highest possible quality of results.

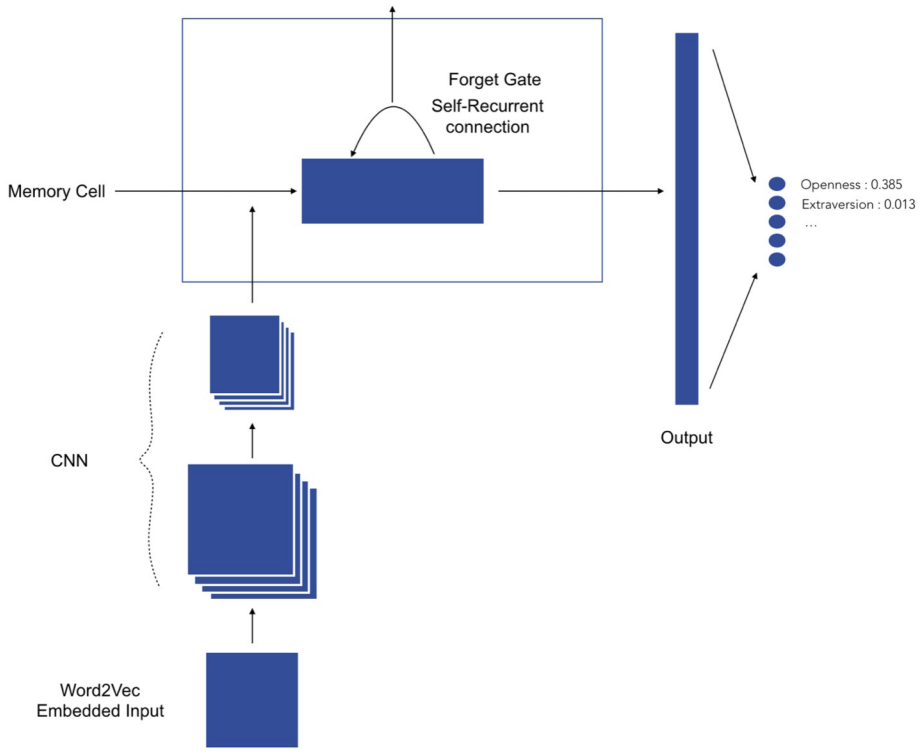
Data

We are using data that was gathered in a study by Pennebaker and King [1999]. It consists of a total of 2,468 daily writing submissions from 34 psychology students (29 women and 5 men whose ages ranged from 18 to 67 with a mean of 26.4). The writing submissions were in the form of a course unrated assignment. For each assignment, students were expected to write a minimum of 20 minutes per day about a specific topic. The data was collected during a 2-week summer course between 1993 to 1996. Each student completed their daily writing for 10 consecutive days. Students' personality scores were assessed by answering the Big Five Inventory (BFI) [John et al., 1991]. The BFI is a 44-item self-report questionnaire that provides a score for each of the five personality traits. Each item consists of short phrases and is rated using a 5-point scale that ranges from 1 (disagree strongly) to 5 (agree strongly). An instance in the data source consists of an ID, the actual essay, and five classification labels of the Big Five personality traits. Labels were originally in the form of either yes ('y') or no ('n') to indicate scoring high or low for a given trait. It is important to note that the classification labels have been applied according to answers to a rather short self-report questionnaire : there might be a non-negligible bias in the data due to both the relative simplicity of the BFI test compared to the complexity of psychological features, and the cognitive biases preventing users from providing a perfectly accurate assessment of their own characteristics.

Requirements

```
Python : 3.6.5

Scipy : 1.1.0

Scikit-learn : 0.20.0

Tensorflow : 1.12.0

Keras : 2.2.2

Numpy : 1.15.2

Nltk : 3.3.0

Gensim : 3.4.0
```

Pipeline



1. The text-based personality recognition pipeline has the following structure :
2. Text data retrieving
3. Custom natural language preprocessing :
    1. Tokenization of the document
    2. Cleaning and standardization of formulations using regular expressions (for instance replacing "can't" by "cannot", "'ve" by "have")
    3. Deletion of the punctuation
    4. Lowercasing the tokens
    5. Removal of predefined stopwords (such as 'a', 'an' etc.)
    6. Application of part-of-speech tags on the remaining tokens
    7. Lemmatization of tokens using part-of-speech tags for more accuracy.
    8. Padding the sequences of tokens of each document to constrain the shape of the input vectors. The input size has been fixed to 300 : all tokens beyond this index are deleted. If the input vector has less than 300 tokens, zeros are added at the beginning of the vector in order to normalize the shape. The dimension of the padded sequence has been determine using the characteristics of our training data. The average number of words in each essay was 652 before any preprocessing. After the standardization of formulations, and the removal of punctuation characters and stopwords, the average number of words dropped to 168 with a

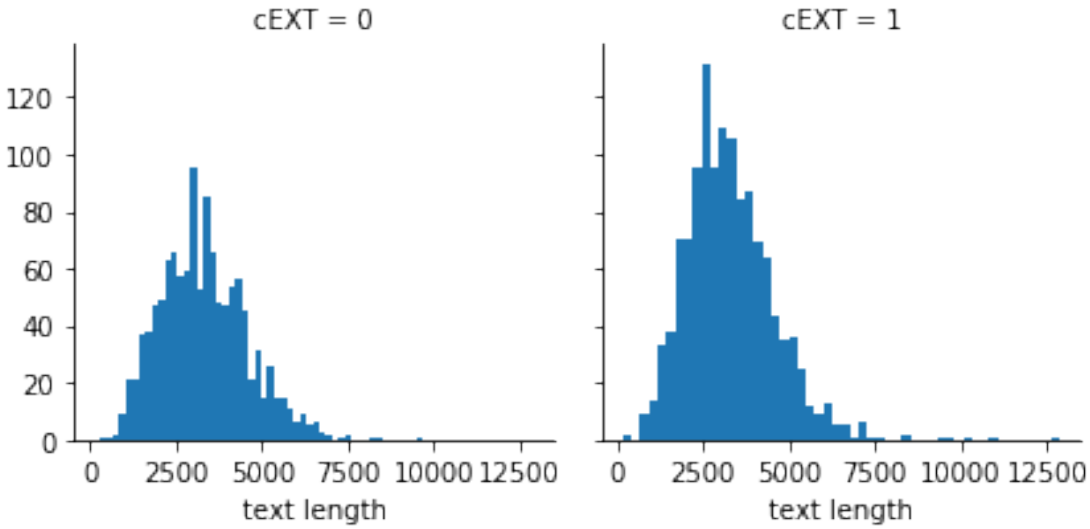
standard deviation of 68. In order to make sure we incorporate in our classification the right number of words without discarding too much information, we set the padding dimension to 300, which is roughly equal to the average length plus two times the standard deviation.

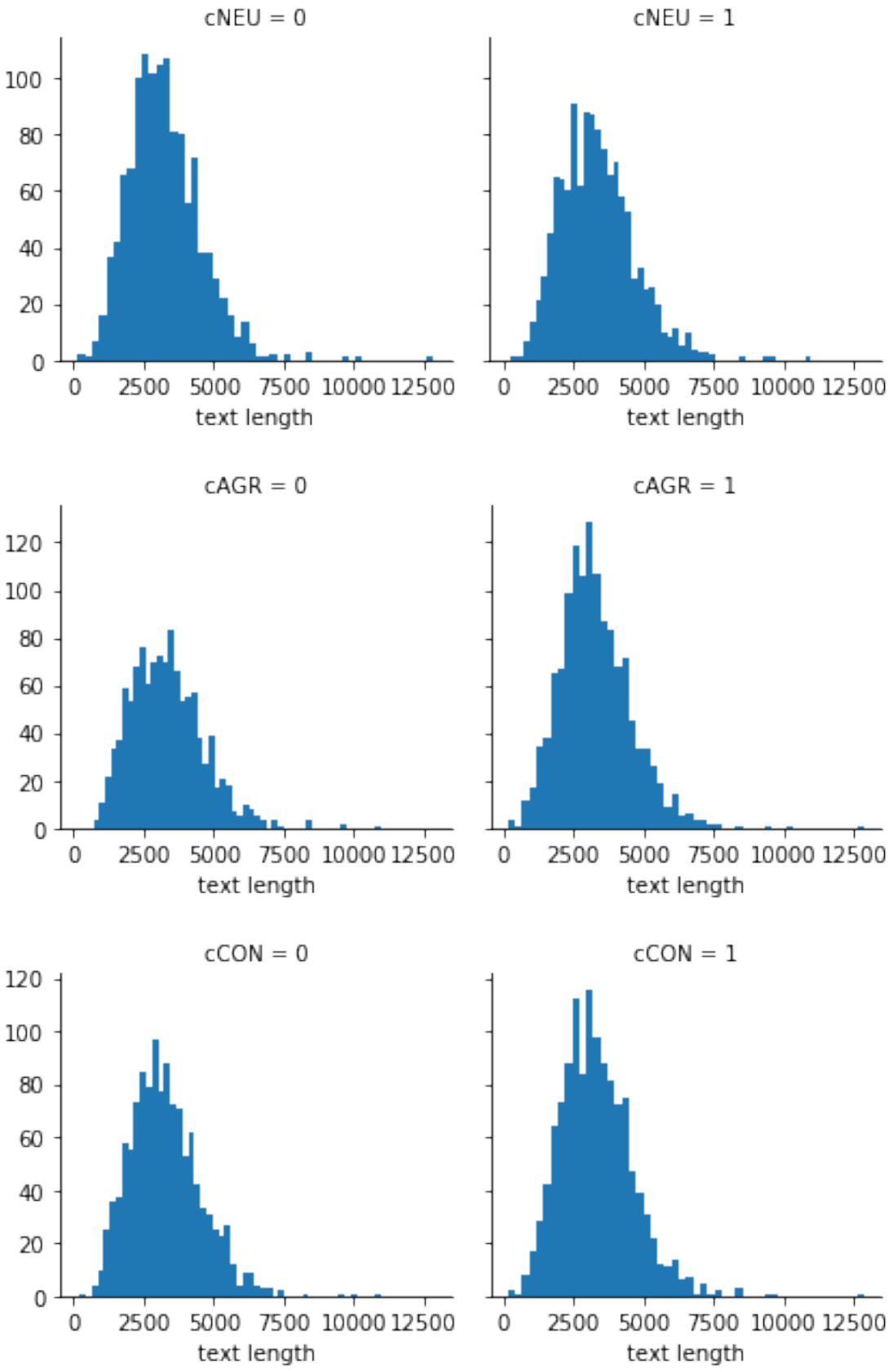
4. 300-dimension Word2Vec trainable embedding

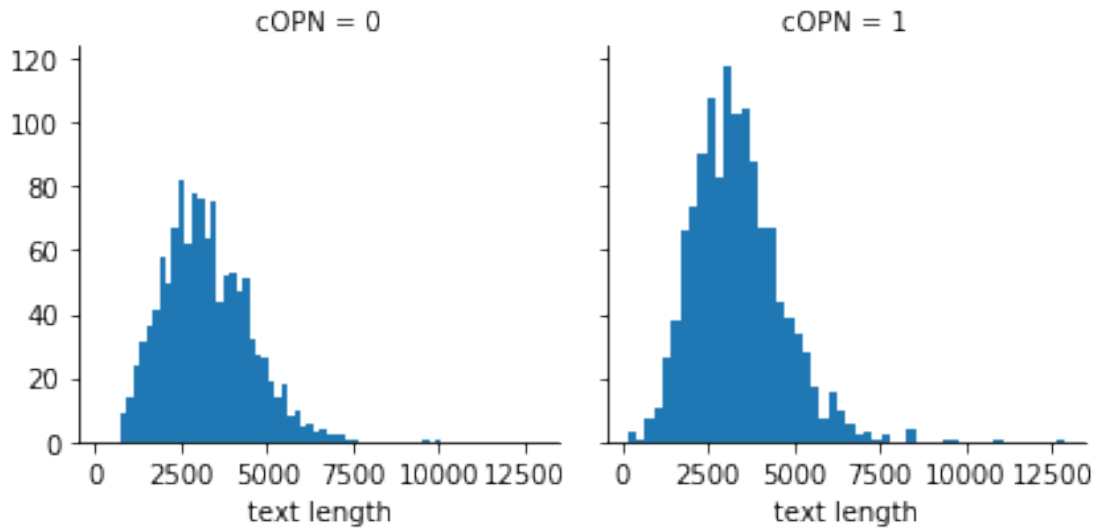
5. Prediction using our pre-trained model

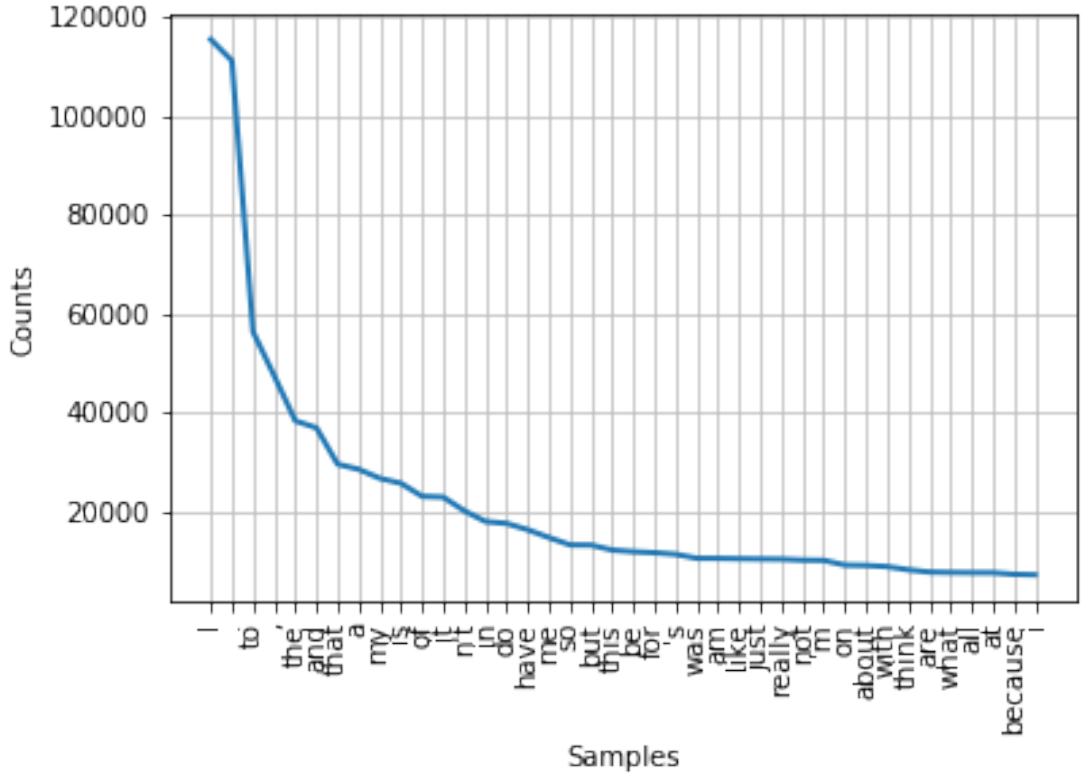
Model

We have chosen a neural network architecture based on both one-dimensional convolutional neural networks and recurrent neural networks. The one-dimensional convolution layer plays a role comparable to feature extraction : it allows finding patterns in text data. The Long-Short Term Memory cell is then used in order to leverage on the sequential nature of natural language : unlike regular neural network where inputs are assumed to be independent of each other, these architectures progressively accumulate and capture information through the sequences. LSTMs have the property of selectively remembering patterns for long durations of time. Our final model first includes 3 consecutive blocks consisting of the following four layers : one-dimensional convolution layer - max pooling - spatial dropout - batch normalization. The numbers of convolution filters are respectively 128, 256 and 512 for each block, kernel size is 8, max pooling size is 2 and dropout rate is 0.3. Following the three blocks, we chose to stack 3 LSTM cells with 180 outputs each. Finally, a fully connected layer of 128 nodes is added before the last classification layer.




cNEU = 0
cNEU = 1
cAGR = 0
cAGR = 1
cCON = 0
cCON = 1
text length
0
2500
5000
7500
10000
12500
0
20
40
60
80
100
120





Train models

```
Building for evaluation
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input1 (InputLayer)          (None, 300)               0
_________________________________________________________________
embedding_15 (Embedding)     (None, 300, 300)          6704100
```

```
_________________________________________________________________
conv1d_38 (Conv1D)           (None, 300, 128)          307328
_________________________________________________________________
max_pooling1d_38 (MaxPooling (None, 150, 128)          0
_________________________________________________________________
spatial_dropout1d_38 (Spatia (None, 150, 128)          0
_________________________________________________________________
batch_normalization_38 (Batc (None, 150, 128)          512
_________________________________________________________________
conv1d_39 (Conv1D)           (None, 150, 256)          262400
_________________________________________________________________
max_pooling1d_39 (MaxPooling (None, 75, 256)           0
_________________________________________________________________
spatial_dropout1d_39 (Spatia (None, 75, 256)           0
_________________________________________________________________
batch_normalization_39 (Batc (None, 75, 256)           1024
_________________________________________________________________
conv1d_40 (Conv1D)           (None, 75, 384)           786816
_________________________________________________________________
max_pooling1d_40 (MaxPooling (None, 37, 384)           0
_________________________________________________________________
spatial_dropout1d_40 (Spatia (None, 37, 384)           0
_________________________________________________________________
batch_normalization_40 (Batc (None, 37, 384)           1536
_________________________________________________________________
lstm_38 (LSTM)               (None, 37, 180)           406800
_________________________________________________________________
lstm_39 (LSTM)               (None, 37, 180)           259920
_________________________________________________________________
lstm_40 (LSTM)               (None, 180)               259920
_________________________________________________________________
dense_24 (Dense)             (None, 128)               23168
_________________________________________________________________
dense_25 (Dense)             (None, 5)                 645
=================================================================
Total params: 9,014,169
Trainable params: 9,012,633
Non-trainable params: 1,536
_________________________________________________________________
```

Performance

We tried different baseline models in order to assess the performance of our final architecture. Here are the accuracies of the different models.

Model	EXT	NEU	AGR	CON	OPN
TF-IDF + MNB	45.34	45.11	45.24	45.31	45.12
TF-IDF + SVM	45.78	45.91	45.41	45.54	45.56
Word2Vec + MNB	45.02	46.01	46.34	46.38	45.97
Word2Vec + SVM	46.18	48.21	49.65	49.97	50.07
Word2Vec (TF-IDF averaging) + MNB	45.87	44.99	45.38	44.21	44.84
Word2Vec (TF-IDF averaging) + SVM	46.01	46.19	47.56	48.11	48.89
Word2Vec + NN (LSTM)	51.98	50.01	51.57	51.11	50.51
Word2Vec + NN (CONV + LSTM)	**55.07**	**50.17**	**54.57**	**53.23**	**53.84**

Facial Emotion Recognition

The aim of this section is to explore facial emotion recognition techniques from a live webcam video stream.

Video Processing

Pipeline

The video processing pipeline was built the following way :

1. Launch the webcam
2. Identify the face by Histogram of Oriented Gradients
3. Zoom on the face
4. Dimension the face to 48 * 48 pixels
5. Make a prediction on the face using our pre-trained model
6. Also identify the number of blinks on the facial landmarks on each picture

Model

The model we have chosen is an XCeption model, since it outperformed the other approaches we developed so far. We tuned the model with :

•Data augmentation

•Early stopping

•Decreasing learning rate on plateau

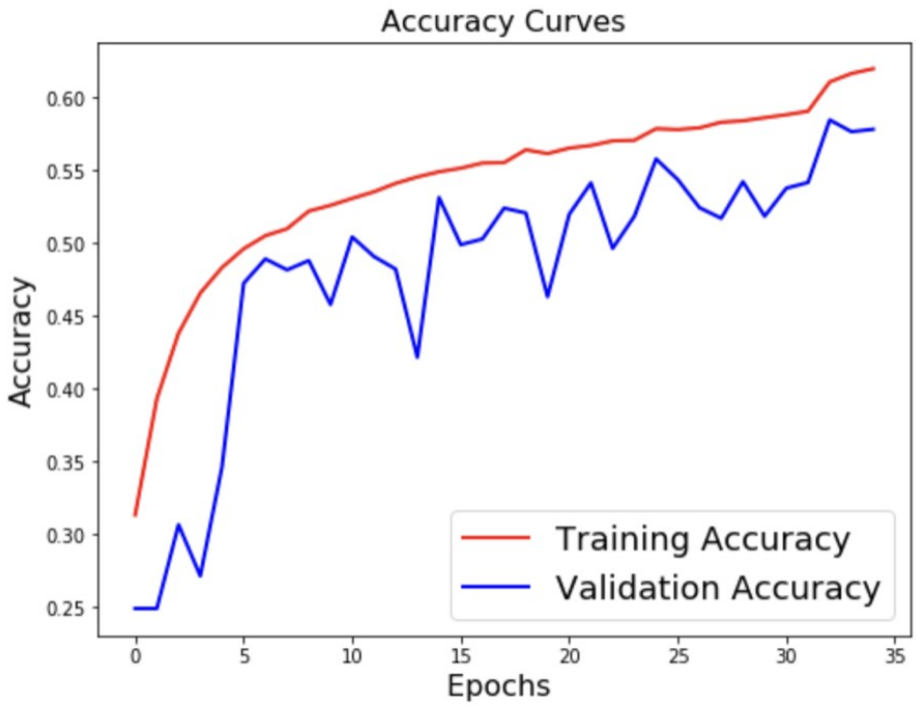
•L2-Regularization

•Class weight balancing

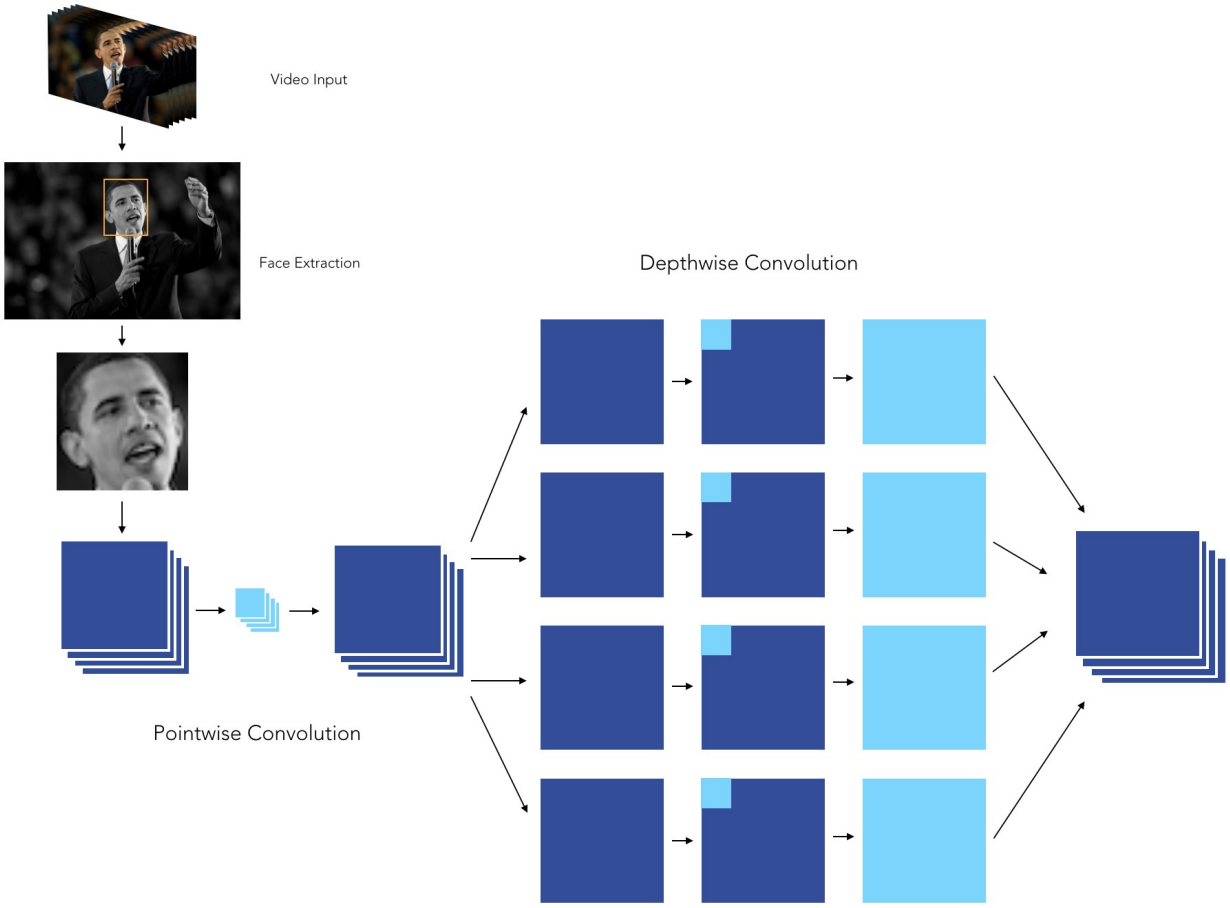
•And kept the best model

As you might have understood, the aim was to limit overfitting as much as possible in order to obtain a robust model.

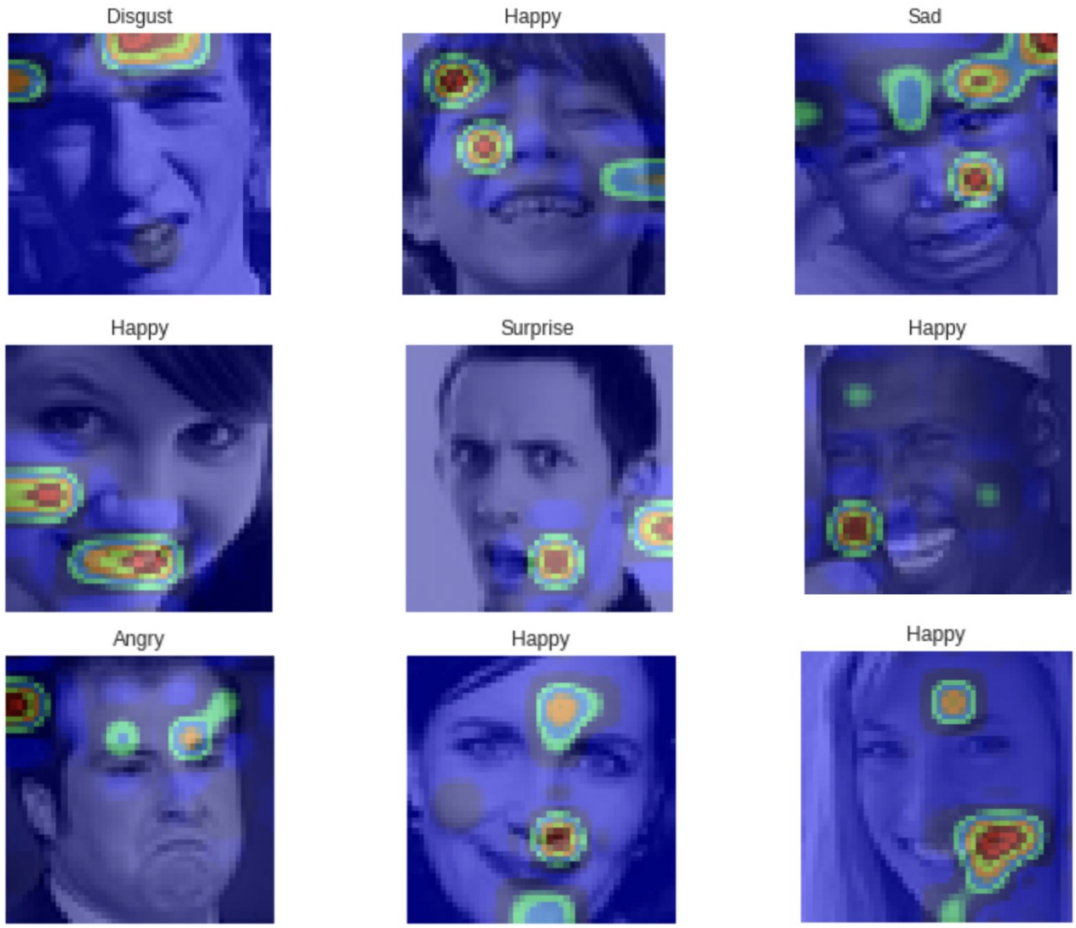
- To know more on how we prevented overfitting, check this article : https://maelfabien.github.io/deeplearning/regu/
- To know more on the XCeption model, check this article : https://maelfabien.github.io/deeplearning/xception/



The XCeption architecture is based on DepthWise Separable convolutions that allow to train much fewer parameters, and therefore reduce training time on Colab's GPUs to less than 90 minutes.



When it comes to applying CNNs in real life application, being able to explain the results is a great challenge. We can indeed plot class activation maps, which display the pixels that have been activated by the last convolution layer. We notice how the pixels are being activated differently depending on the emotion being labeled. The happiness seems to depend on the pixels linked to the eyes and mouth, whereas the sadness or the anger seem for example to be more related to the eyebrows.



Notebooks

Among the notebooks, the role of each notebook is the following :

- `01-Pre-Processing.ipynb` : Transform the initial CSV file into train and test data sets
- `02-HOG_Features.ipynb` : A manual extraction of features (Histograms of Oriented Gradients, Landmarks) and SVM
- `03-Pre-Processing-EmotionalDAN.ipynb` : An implementation of Deep Alignment Networks to extract features
- `04-LGBM.ipynb` : Use of classical Boosting techniques on top on flatenned image or auto-encoded image
- `05-Simple_Arch.ipynb` : A simple Deep Learning Architecture
- `06-Inception.ipynb` : An implementation of the Inception Architecture
- `07-Xception.ipynb` : An implementation of the Xception Architecture
- `08-DeXpression.ipynb` : An implementation of the DeXpression Architecture
- `09-Prediction.ipynb` : Live Webcam prediction of the model
- `10-Hybrid.ipynb` : A hybrid deep learning model taking both the HOG/Landmarks model and the image

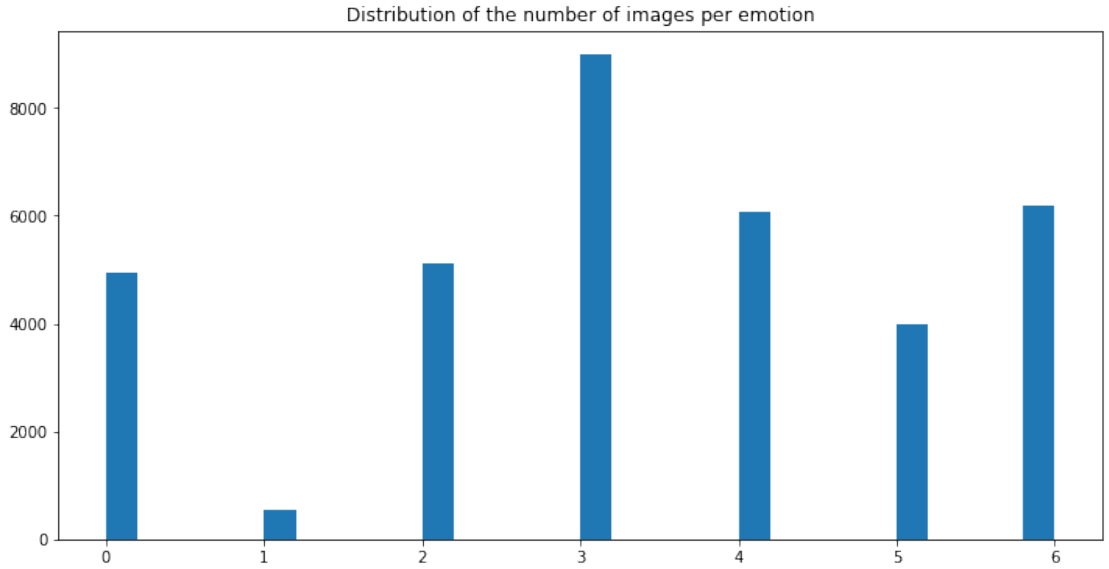
Instead of laying all the Notebooks one by one in front of you, it would be better if you could better able to see all in one go. So here is everything.

Context

The models explored include :

- Manual filters
- Deep Learning Architectures
- DenseNet Inspired Architectures

This model will be combined with voice emotion recongition as well as psychological traits extracted from text inputs, and should provide a benchmark and a deep analysis of both verbal and non-verbal insights for candidates seeking for a job and their performance during an interview.



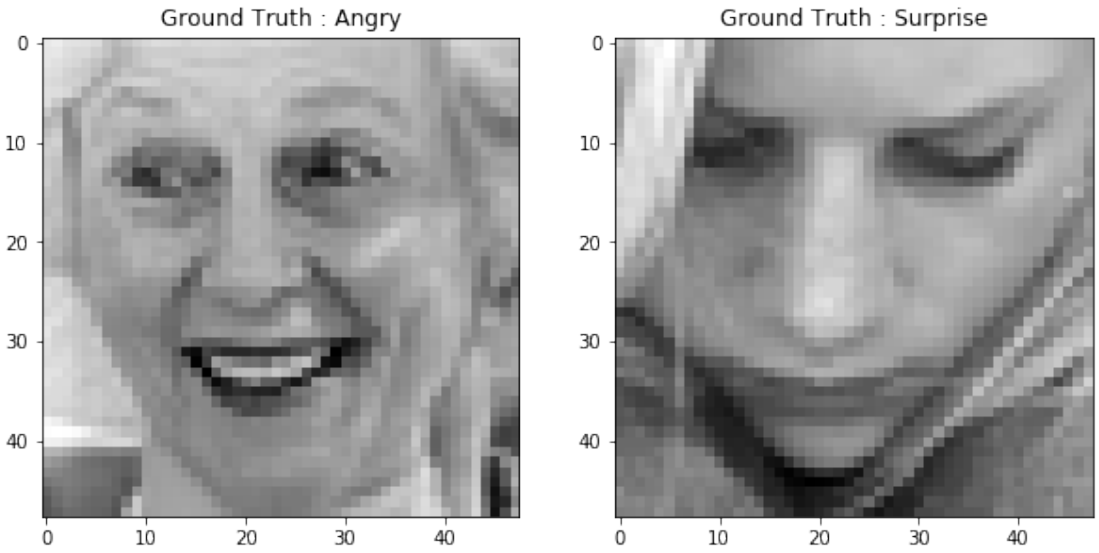

```
train.shape
```

```
(28709, 3)
```

```
test.shape
```

```
(3589, 3)
```

```
(Text(0.5, 1.0, 'Ground Truth : Surprise')
```



VI. Detect Faces

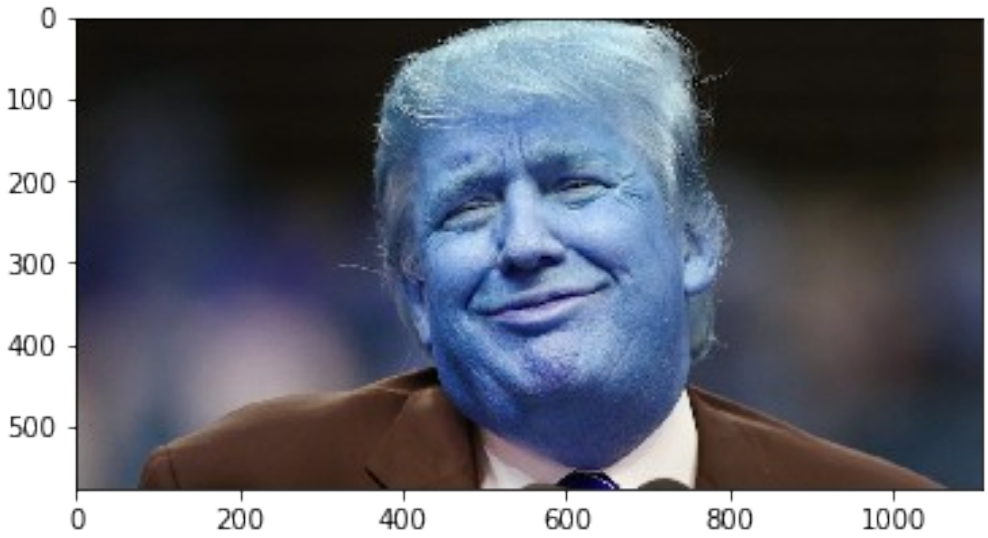
First of all, we need to detect the faces inside an image. This will allow us to :

- focus on the region of the face
- stop the prediction if no face is recognized.

To do so, we use OpenCV faceCascade classifier. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

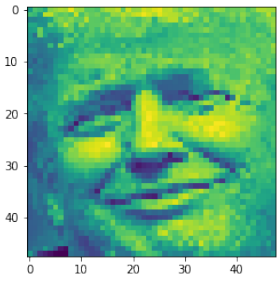
The term "Cascade" comes from the fact that when a window is explored and no face edge is identified, the region is left apart and we move on to the next one using Adaboost classifier. This makes the overall process very efficient.

```
<matplotlib.image.AxesImage at 0x1a32e6e6a0>
```



Extracted face :

```
face = extract_face_features(detect_face(trump_face))[0]
plt.imshow(face)
```




```
<matplotlib.image.AxesImage at 0x1a32ec6dd8>
```

Build Model

```
model = createModel3()
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 20)        200
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 30)        5430
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 30)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 30)        120
```

```
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 30)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 40)        10840
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 50)        18050
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 50)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 50)        200
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 50)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 60)        27060
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 70)        37870
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 70)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 70)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 80)          50480
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 90)          64890
_________________________________________________________________
flatten_1 (Flatten)          (None, 3240)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              3241000
_________________________________________________________________
dense_2 (Dense)              (None, 512)               512512
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 3591
=================================================================
Total params: 3,972,243
Trainable params: 3,972,083
Non-trainable params: 160
_________________________________________________________________
```

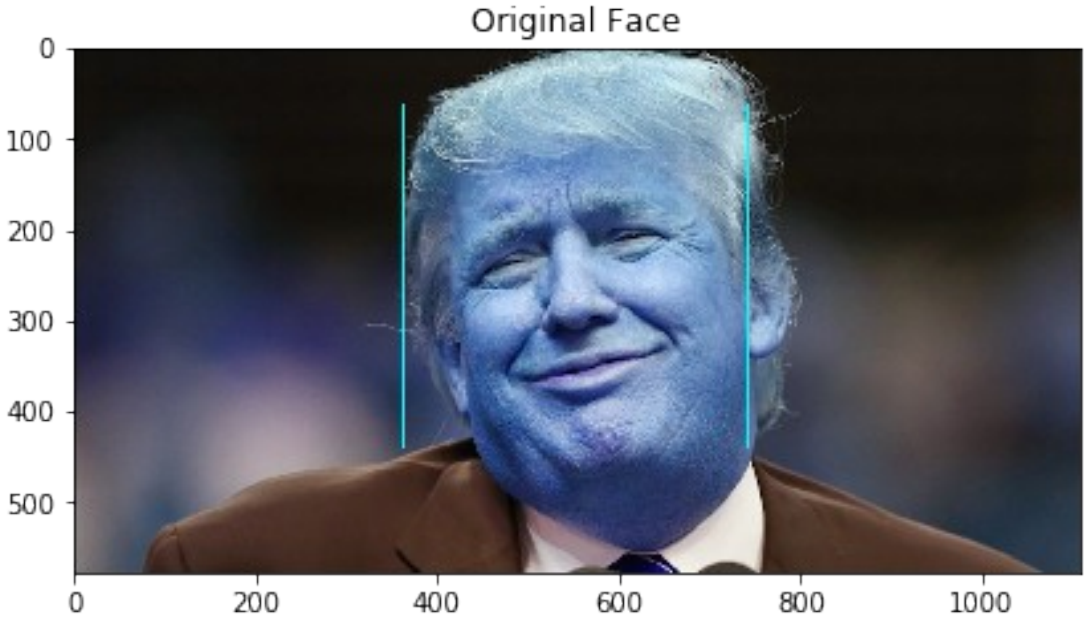
And visualize the model architecture :

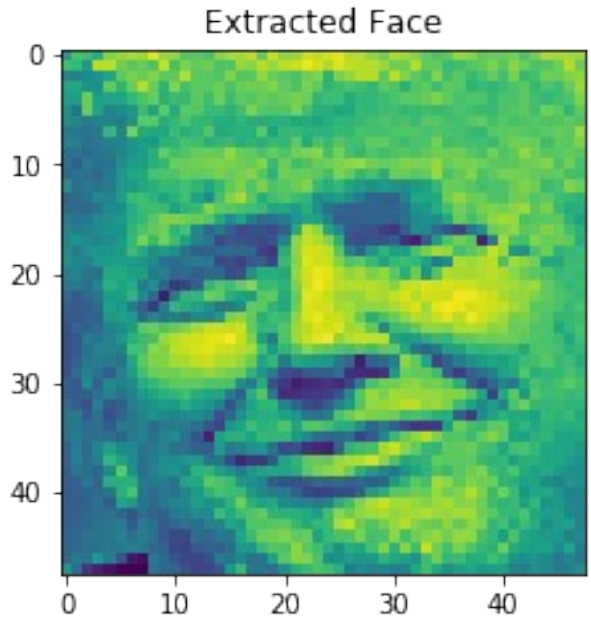
```
plot_model(model, to_file='model_images/model_plot.png', show_shapes=True,
show_layer_names=True)
```

VIII. Visualize layers and output

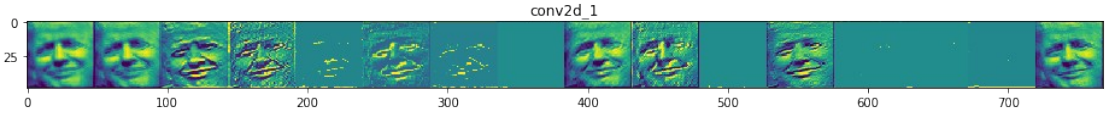
```
layer_outputs = [layer.output for layer in model.layers[:12]]
# Extracts the outputs of the top 12 layers
```

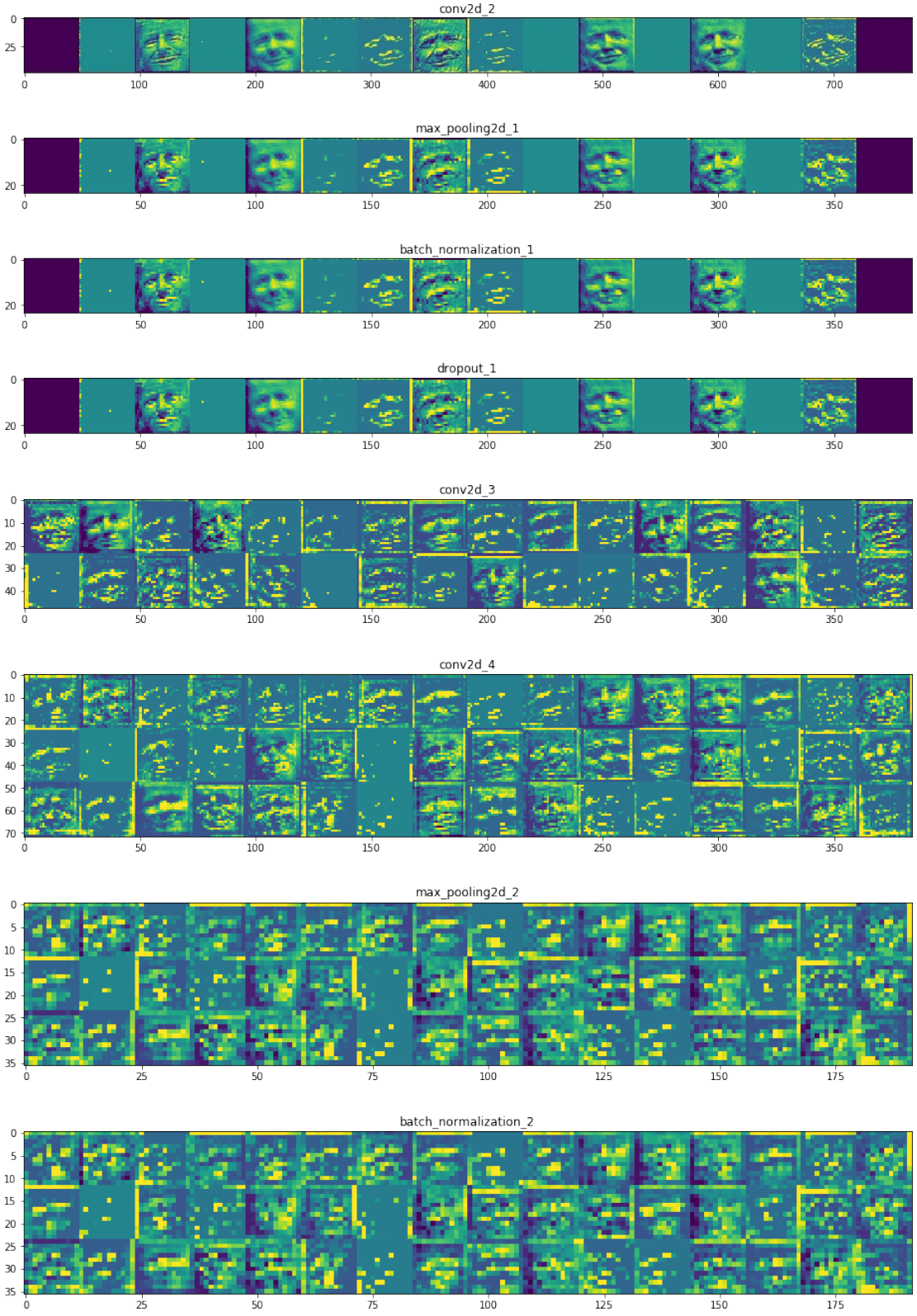
```
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)

layer_names = []
for layer in model.layers[:12]:
    layer_names.append(layer.name) # Names of the layers, so you can have them as part of your plot

images_per_row = 16

trump = '/Users/maelfabien/filrouge_pole_emploi/Video/test_samples/trump.jpg'
trump_face = cv2.imread(trump)
face = extract_face_features(detect_face(trump_face))[0]

to_predict = np.reshape(face.flatten(), (1,48,48,1))
res = model.predict(to_predict)
activations = activation_model.predict(to_predict)

plt.figure(figsize=(12,8))

plt.subplot(211)
plt.title("Original Face")
plt.imshow(trump_face)

plt.subplot(212)
plt.title("Extracted Face")
plt.imshow(face)

plt.show()
```

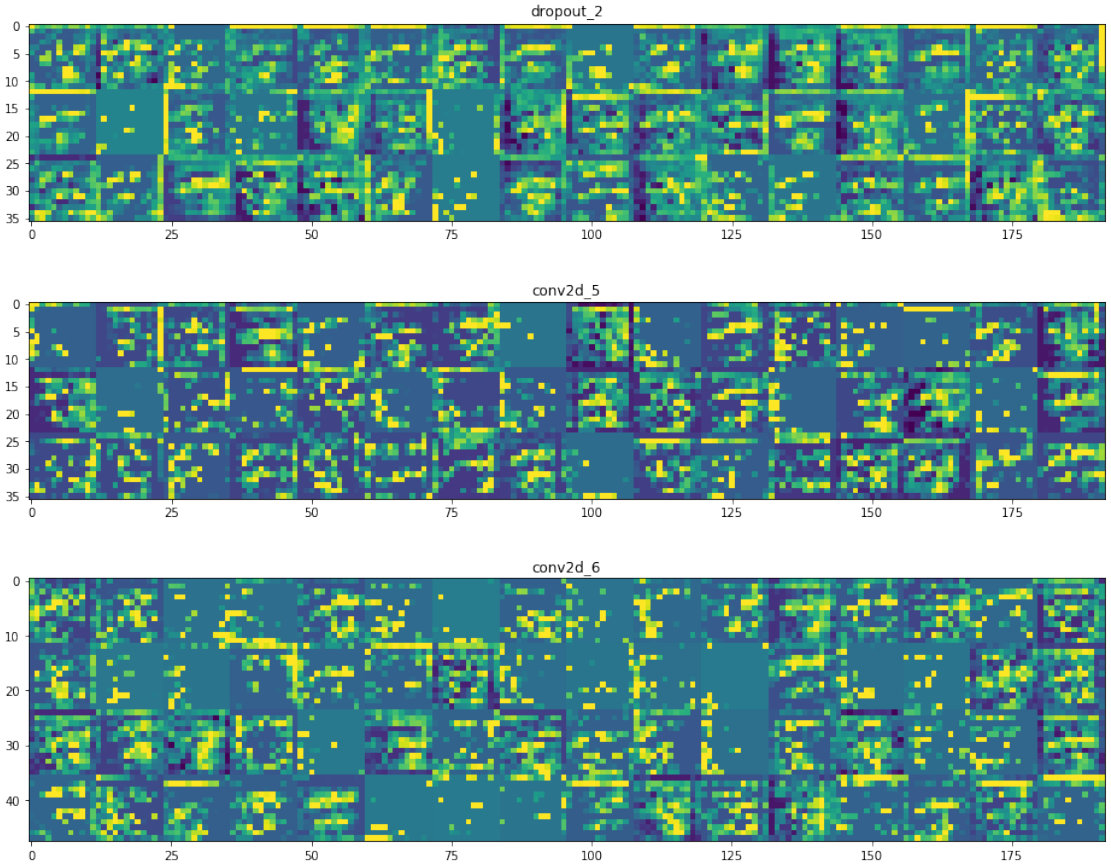


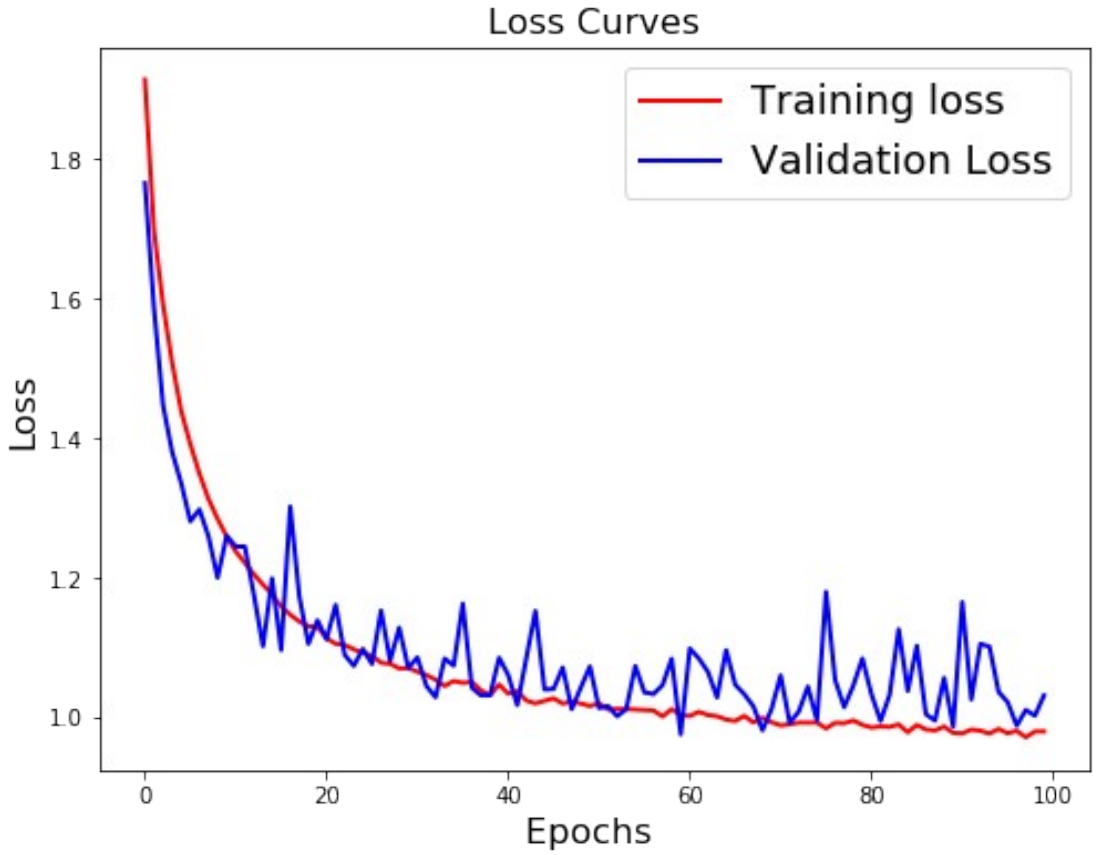


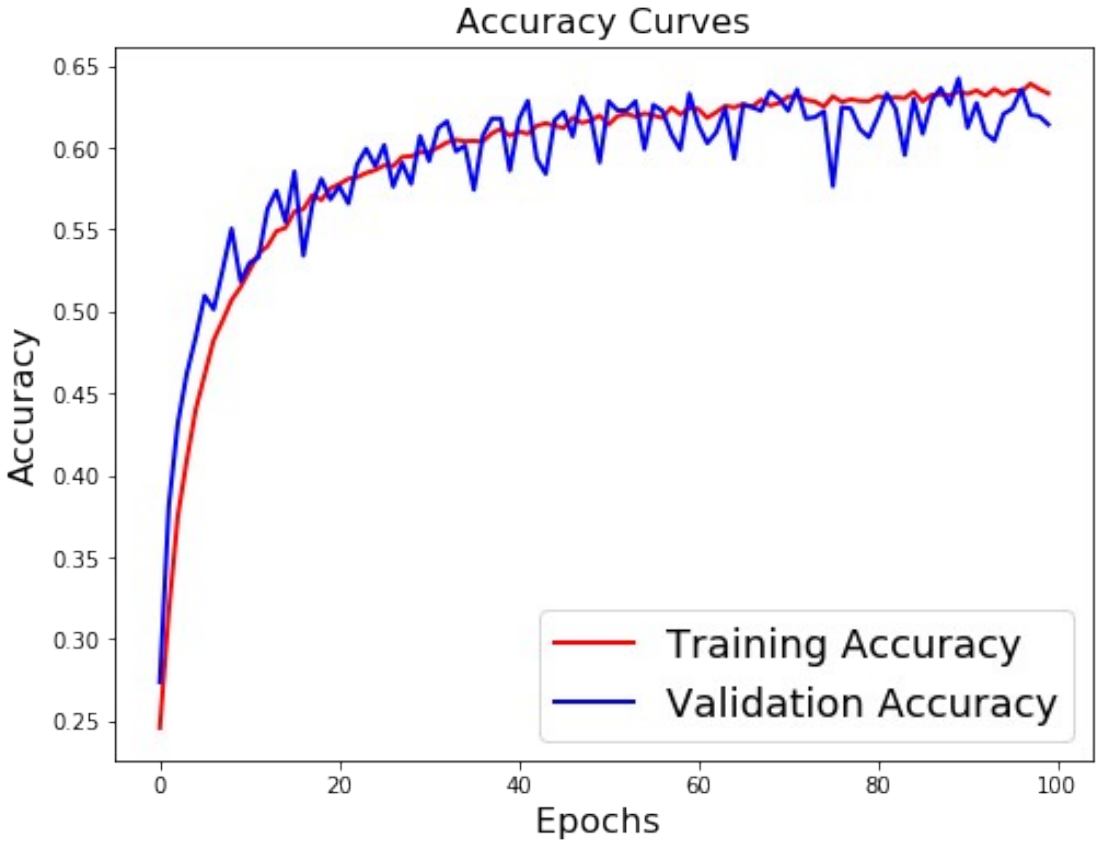

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:10:
RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
```




conv2d_2
max_pooling2d_1
batch_normalization_1
dropout_1
conv2d_3
conv2d_4
max_pooling2d_2
batch_normalization_2


dropout_2
conv2d_5
conv2d_6


Loss Curves
Training loss
Validation Loss
Loss
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100
Epochs


Accuracy Curves
Accuracy
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
Training Accuracy
Validation Accuracy
0
20
40
60
80
100
Epochs

XI. Save and re-open the model

```
#save the model weights
json_string = model.to_json()
model.save_weights(local_path + 'savedmodels/model_3.h5')
open(local_path + 'savedmodels/model_3.json', 'w').write(json_string)
#model.save_weights(local_path + 'savedmodels/Emotion_Face_Detection_Model.h5')
```

```
8372
```

```
with open(local_path + 'savedmodels/model_3.json','r') as f:
    json = f.read()
model = model_from_json(json)


model.load_weights(local_path + 'savedmodels/model_3.h5')
print("Loaded model from disk")
```

```
Loaded model from disk
```

XII. Making a prediction on an image

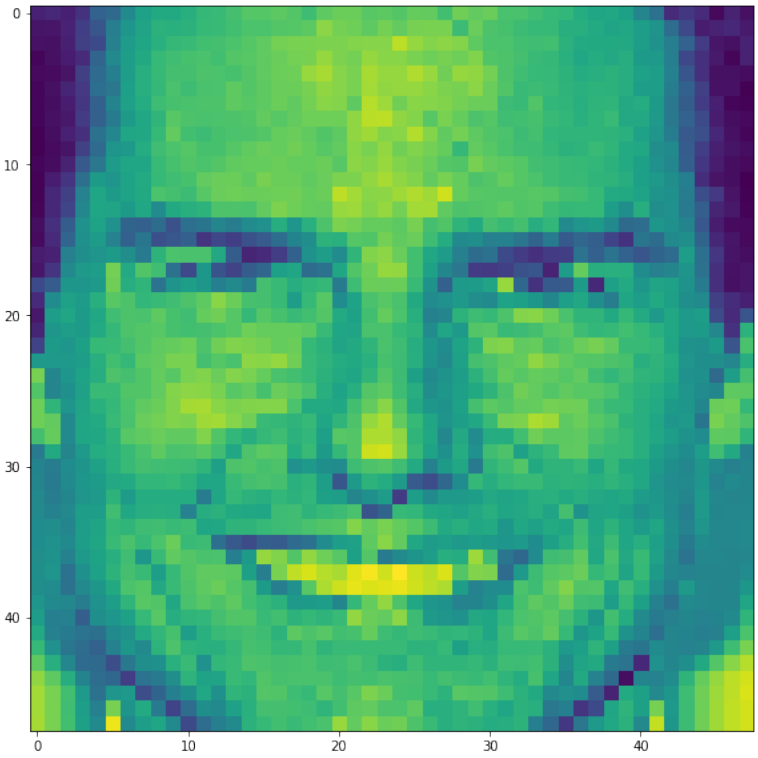
```
hanks = '/Users/maelfabien/filrouge_pole_emploi/Video/test_samples/hanks_vs.jpg'
hanks_face = cv2.imread(hanks)

plt.imshow(cv2.cvtColor(hanks_face, cv2.COLOR_BGR2RGB))
```

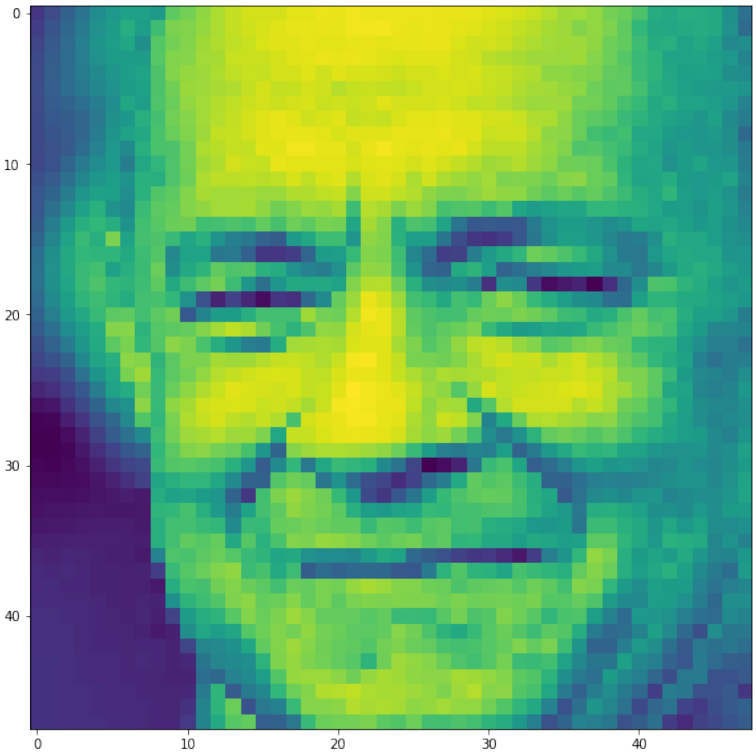
```
<matplotlib.image.AxesImage at 0x1a3b7ee358>
```


```
plt.figure(figsize=(12,12))
plt.imshow(detect_face(hanks_face)[0])
```

```
plt.show()
```


```
for face in extract_face_features(detect_face(hanks_face)) :
    plt.figure(figsize=(10,10))
    plt.imshow(face)
    plt.show()
```






```
for face in extract_face_features(detect_face(hanks_face)) :
    to_predict = np.reshape(face.flatten(), (1,48,48,1))
    res = model.predict(to_predict)
    result_num = np.argmax(res)
```

```
print(result_num)
```

```
3
3
```

This corresponds to the Happy Labels which is a good prediction.

Enhanced Visualization

This basic step is now woring properly and results are quite satisfying. There are lots of sources of improvements we'll try to implement over time :

- add features from manually selected filters (e.g Gabor filters)
- take into account the frequency of eye blinks
- take into account the symetry of the keypoints on a face
- display all the keypoints of the face
- align the face by scaling of the facial features
- add emojis translating the emotion

a. Frequency of eye blink

b. Detect Keypoints to plot them

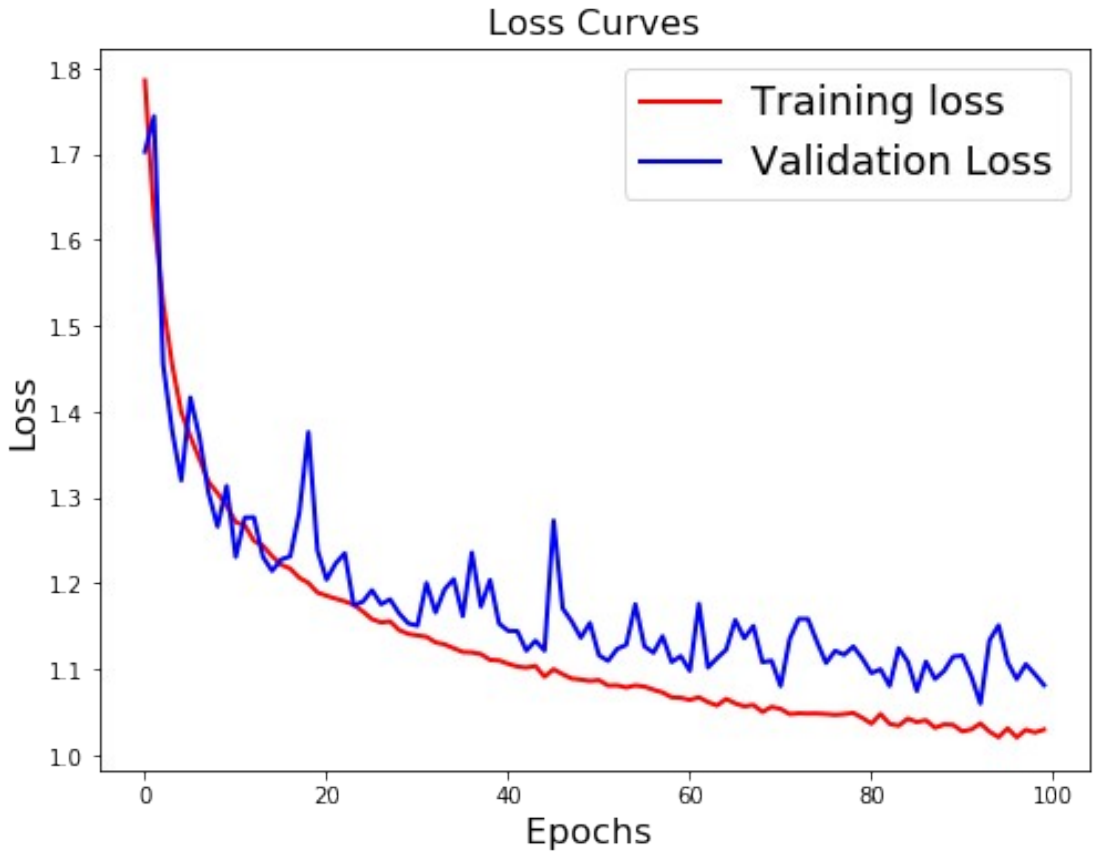
c. Face Alignment

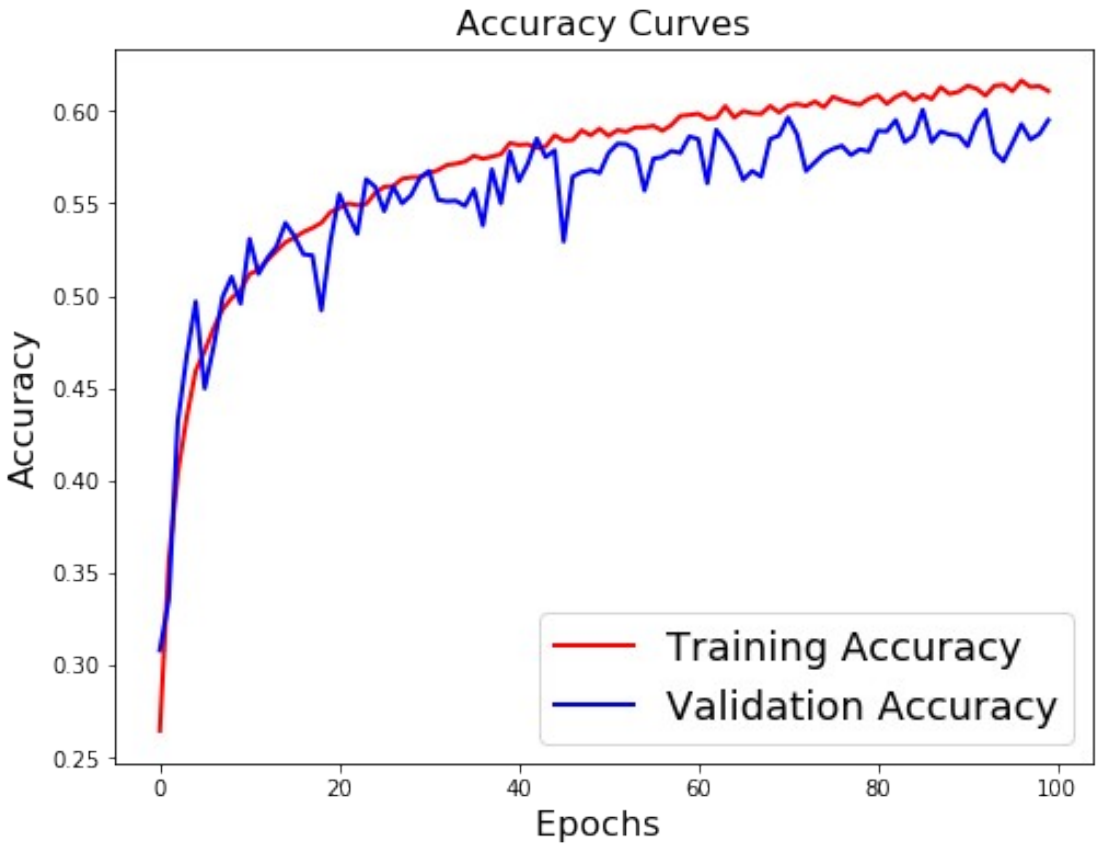
d. Final Prediction

Deeper CNN Models

Inception

Architecture Inspired by : https://ieeexplore.ieee.org/document/7477450





Sources

- Visualization : https://github.com/JostineHo/mememoji/blob/master/data_visualization.ipynb
- State of the art Architecture : https://github.com/amineHorseman/facial-expression-recognition-using-cnn
- Eyes Tracking : https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
- Face Alignment : https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/
- C.Pramerdorfer, and M.Kampel.Facial Expression Recognition using Con-volutional Neural Networks: State of the Art. Computer Vision Lab, TU Wien. https://arxiv.org/pdf/1612.02903.pdf
- A Brief Review of Facial Emotion Recognition Based on Visual Information : https://www.mdpi.com/1424-8220/18/2/401/pdf
- Going deeper in facial expression recognition using deep neural networks : https://ieeexplore.ieee.org/document/7477450

Performance

The models have been trained on Google Colab using free GPUs.The set of emotions we are trying to predict are the following :

1. Happiness 2. Sadness 3. Fear 4. Disgust

5. Surprise 6. Neutral 7. Anger

Features	Accuracy
LGBM on flat image	**--.-%**
LGBM on auto-encoded image	**--.-%**
SVM on HOG Features	**32.8%**
SVM on Facial Landmarks features	**46.4%**
SVM on Facial Landmarks and HOG features	**47.5%**
SVM on Sliding window Landmarks & HOG	**24.6%**
Simple Deep Learning Architecture	**62.7%**
Inception Architecture	**59.5%**
Xception Architecture	**64.5%**
DeXpression Architecture	**--.-%**
Hybrid (HOG, Landmarks, Image)	**45.8%**

Multimodal Emotion Recognition WebApp

We analyse facial, vocal and textual emotions, using mostly deep learning based approaches. We deployed a web app using Flask .

Procedure to Run

To use the web app :

1. Clone the project locally
2. Go in the WebApp folder
3. Run `$ pip install -r requirements.txt``
4. Launch the app by running `python main.py`
5. Go to http://127.0.0.1:5000/ (if running as the local host or else follow the link given in your terminal or command prompt)

Getting the feedback

For both the text and the audio, a button will directly allow you to get feedback. A dashboard displays your performance compared to other candidates.

For the video, due to restrictions of Flask, we are recording the video input for 45 seconds. After this time, the image will freeze. Simply switch the URL to `/video_dash` instead of `/video1` in the URL bar to go to the dashboard.

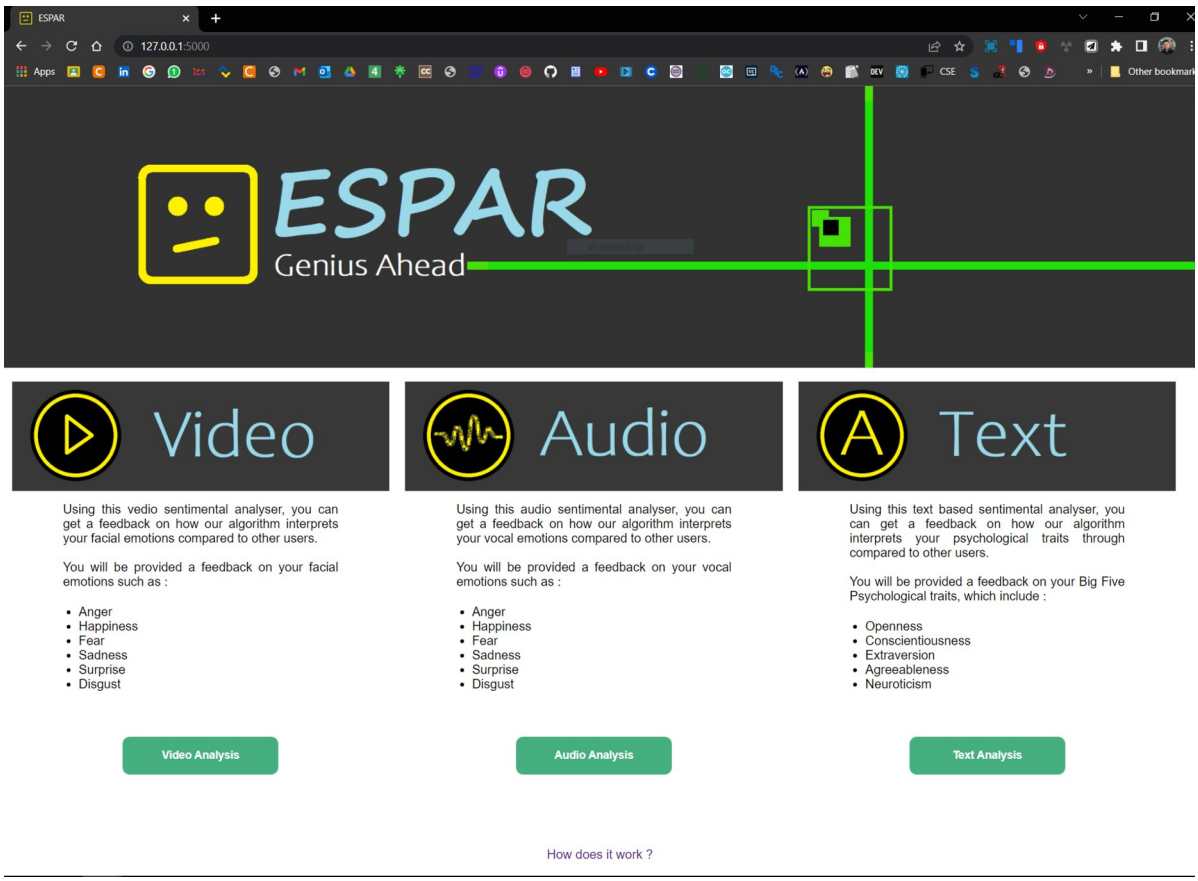
Organization of Files and Folder

The organization of the project is the following :

- •Models : All the pre-trained models used by the WebApp
- •library : The Python scripts that run the emotion detection algorithms
- •static :
  - •CSS : The CSS style sheet and fixed images to display
  - •JS : The JavaScript of the app (D3.js) and the databases that store the information
- •templates : All the HTML pages of the project
- •tmp : Temporary files (i.e. an image from video interview, an audio file or a PDF)
- •main.py : The Flask page that calls the functions and redirects to HTML files

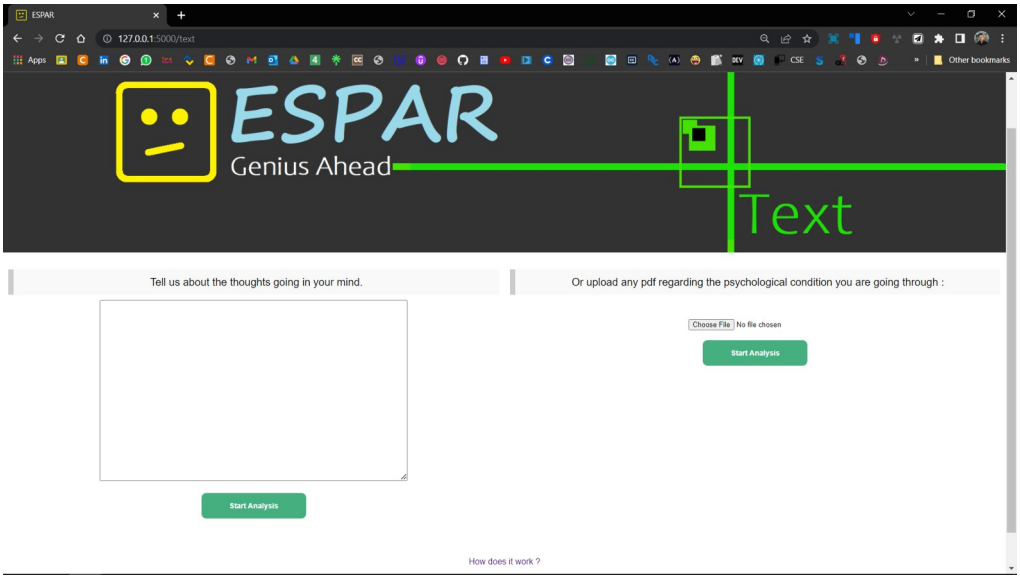
Flask Server Program: Sample Run

The home Screen or home web-page

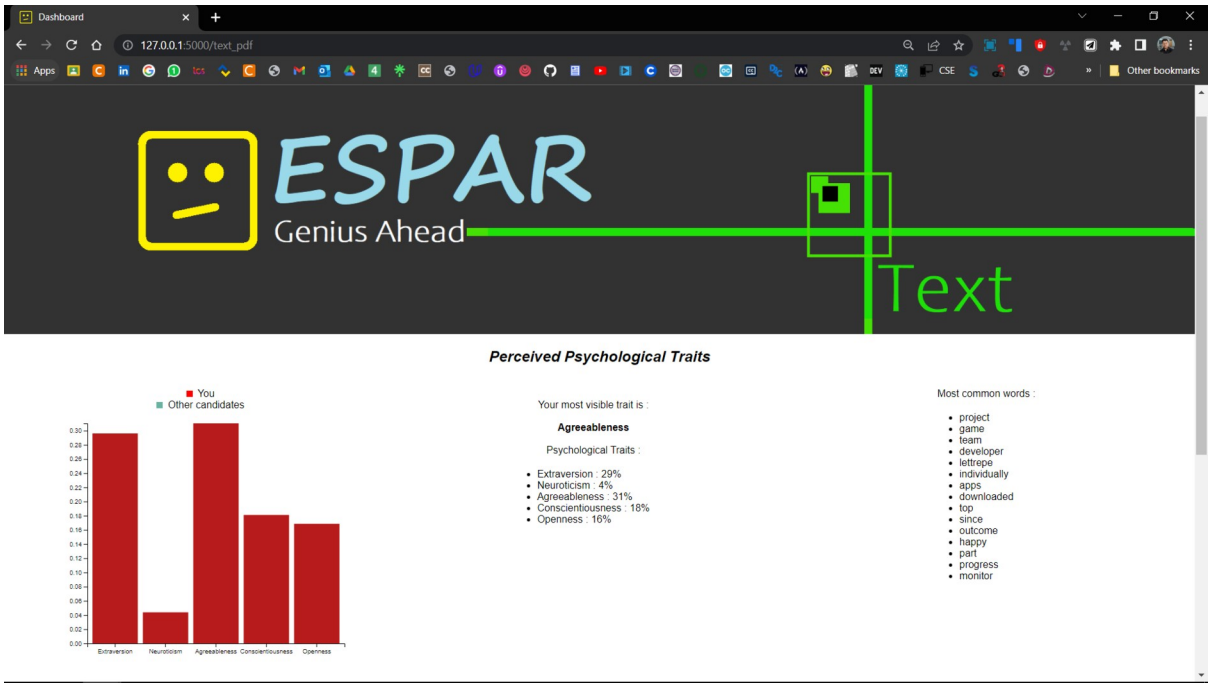


1. TEXT ANALYSIS

http://127.0.0.1:5000/text

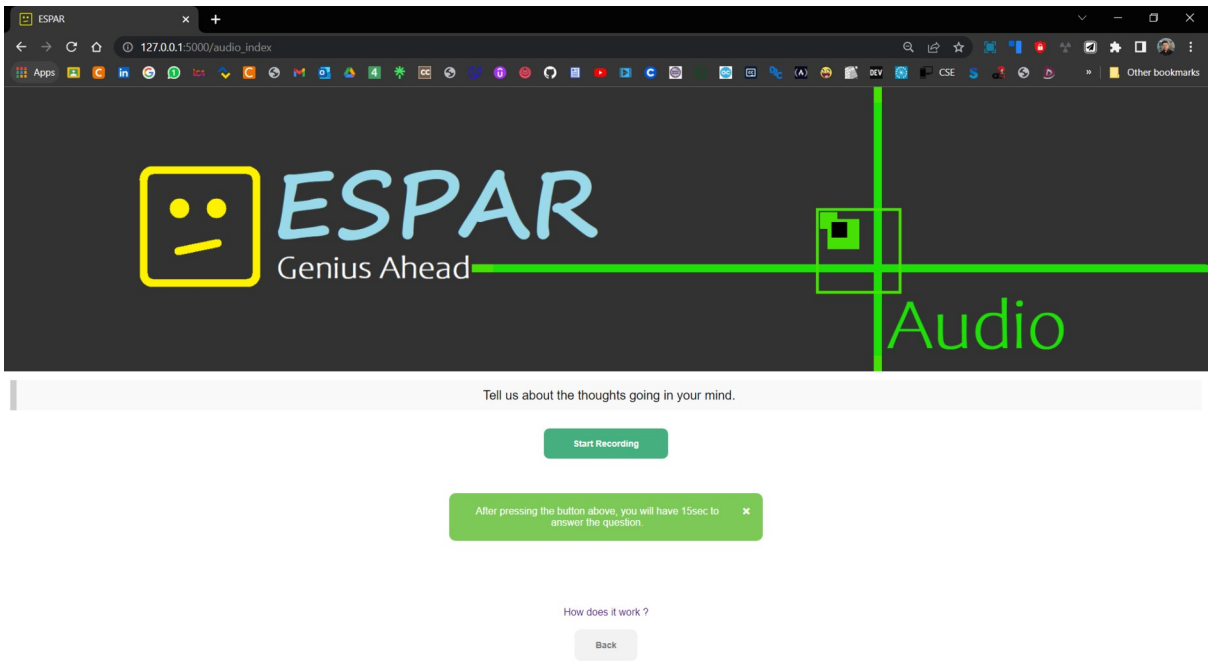


TEXT ANALYSIS : http://127.0.0.1:5000/text_pdf

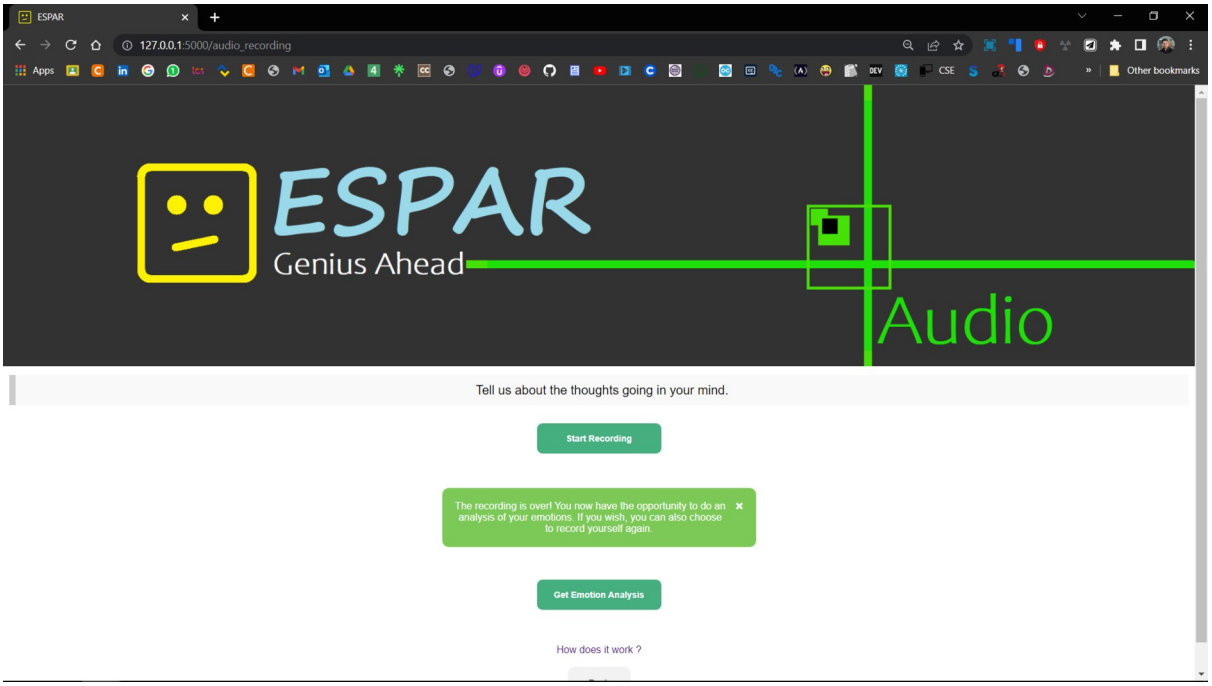


2. AUDIO ANALYSIS

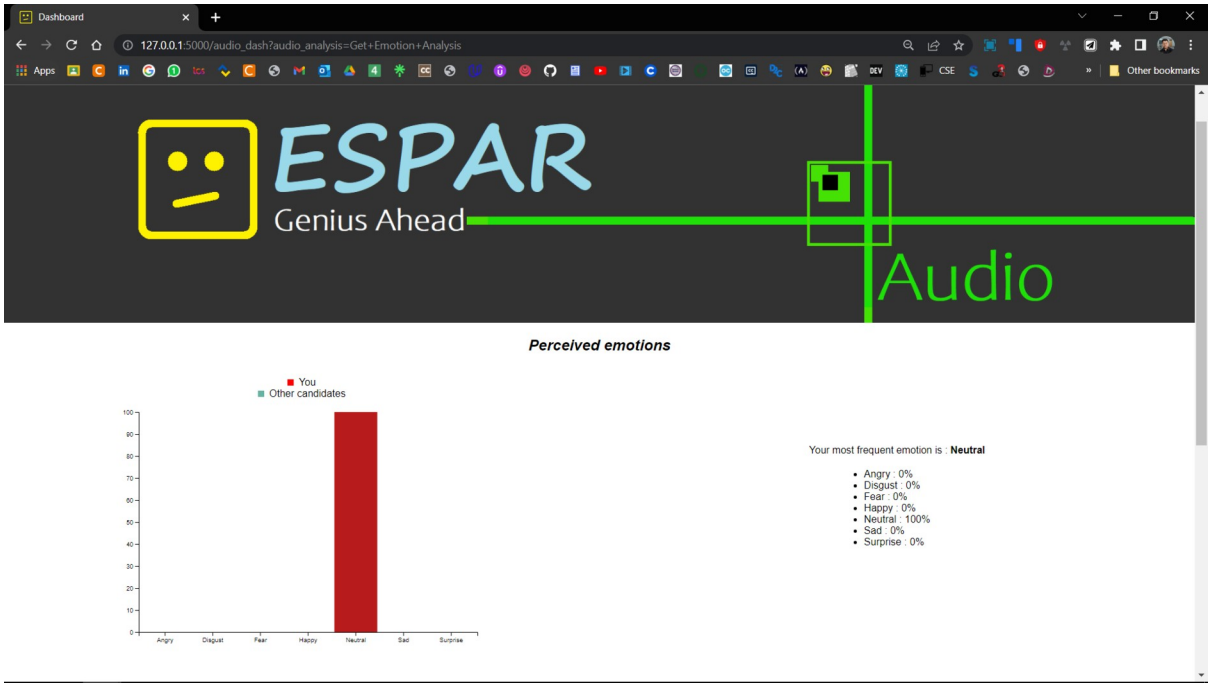
http://127.0.0.1:5000/audio_index



After the recording has started and finished : http://127.0.0.1:5000/audio_recording

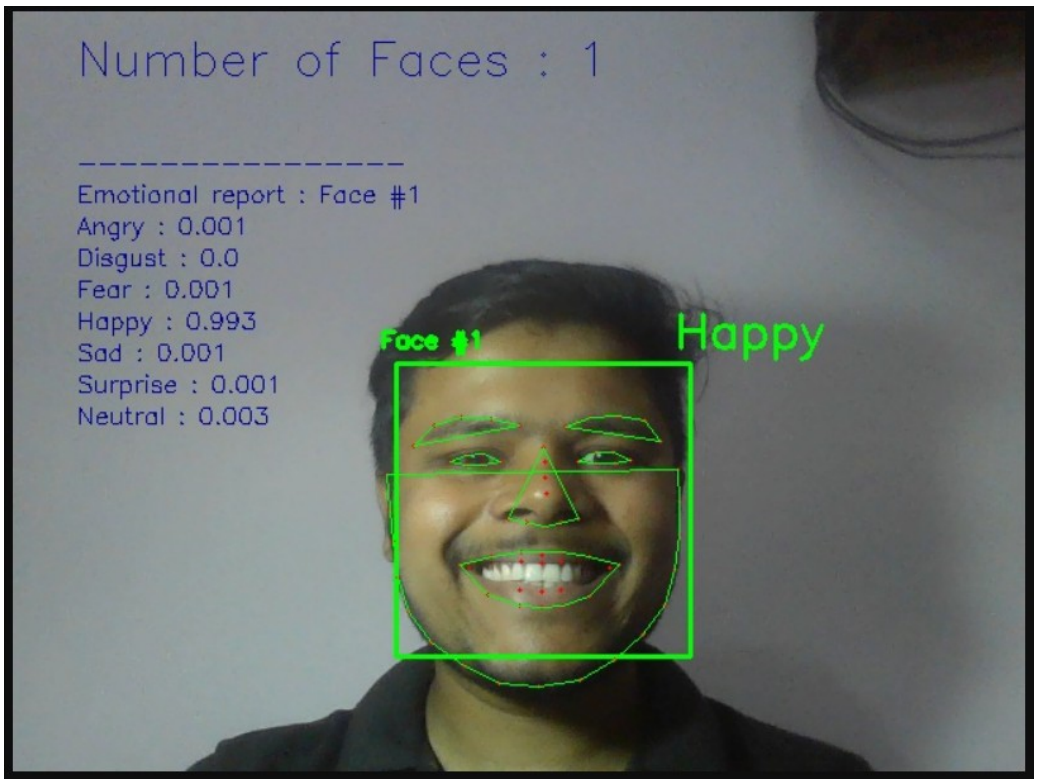


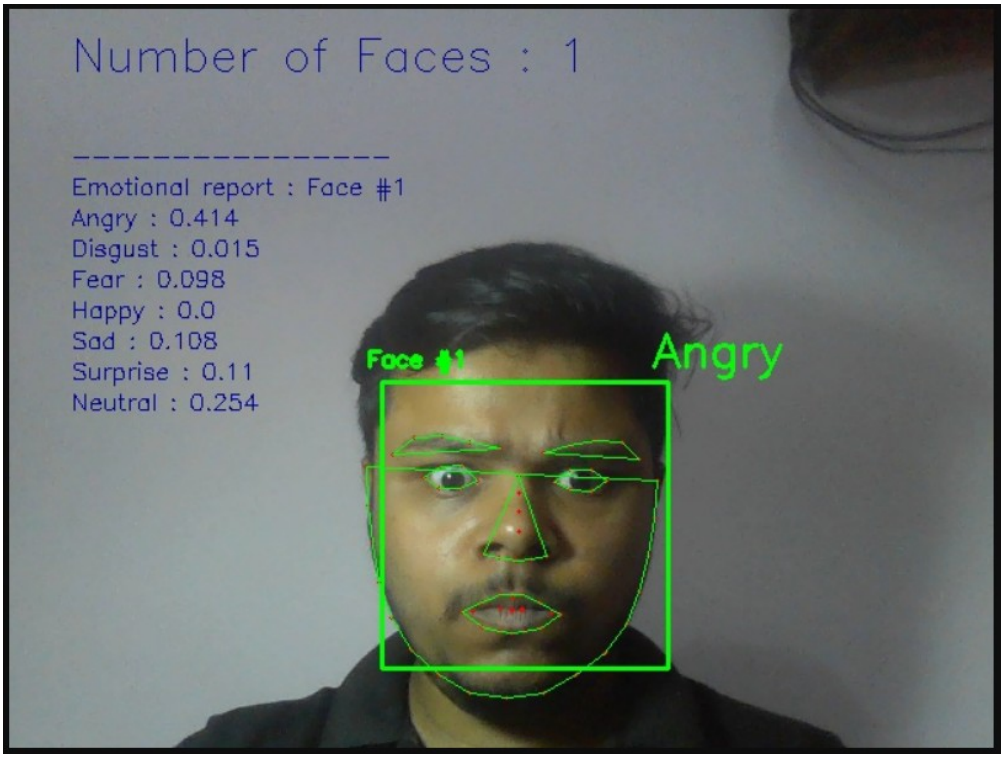
Audio Analysis Results: http://127.0.0.1:5000/audio_dash?audio_analysis=Get+Emotion+Analysis

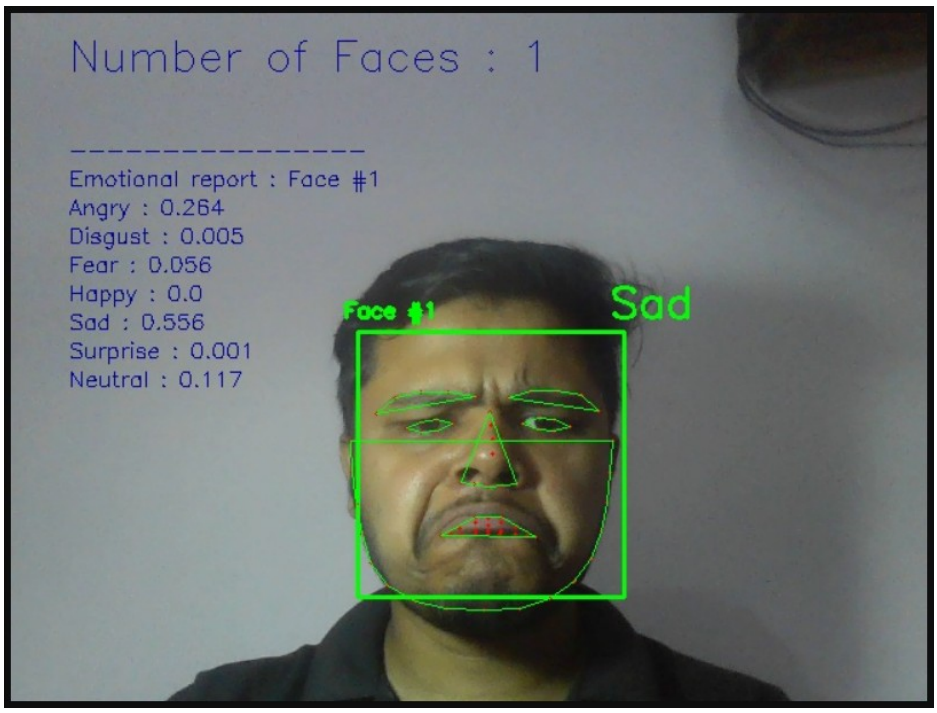


3. VIDEO SENTIMENTAL ANALYSIS

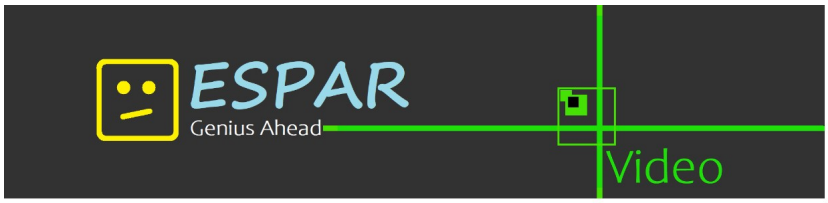
http://127.0.0.1:5000/video_1



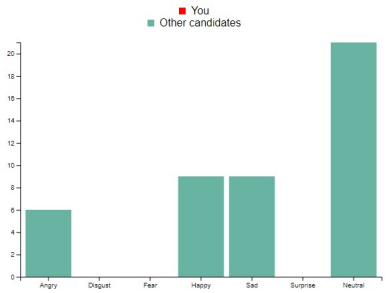



Number of Faces : 1
Emotional report : Face #1
Angry : 0.264
Disgust : 0.005
Fear : 0.056
Happy : 0.0
Sad : 0.556
Surprise : 0.001
Neutral : 0.117
Face #1
Sad

Video Analysis Results : http://127.0.0.1:5000/video_dash



Perceived emotions



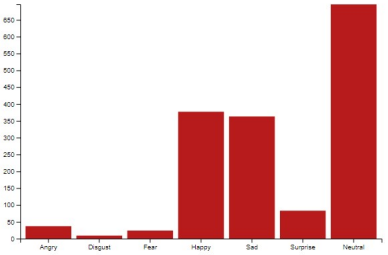
Facial Emotions

Your most frequent emotion is :

Neutral

- Anger : 13%
- Disgust : 0%
- Fear : 0%
- Happiness : 20%
- Sadness : 20%
- Surprise : 0%
- Neutrality : 46%

Other candidates

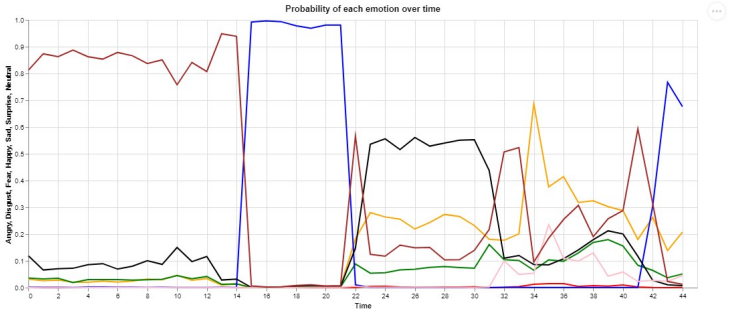


Other candidates most frequent emotion is :

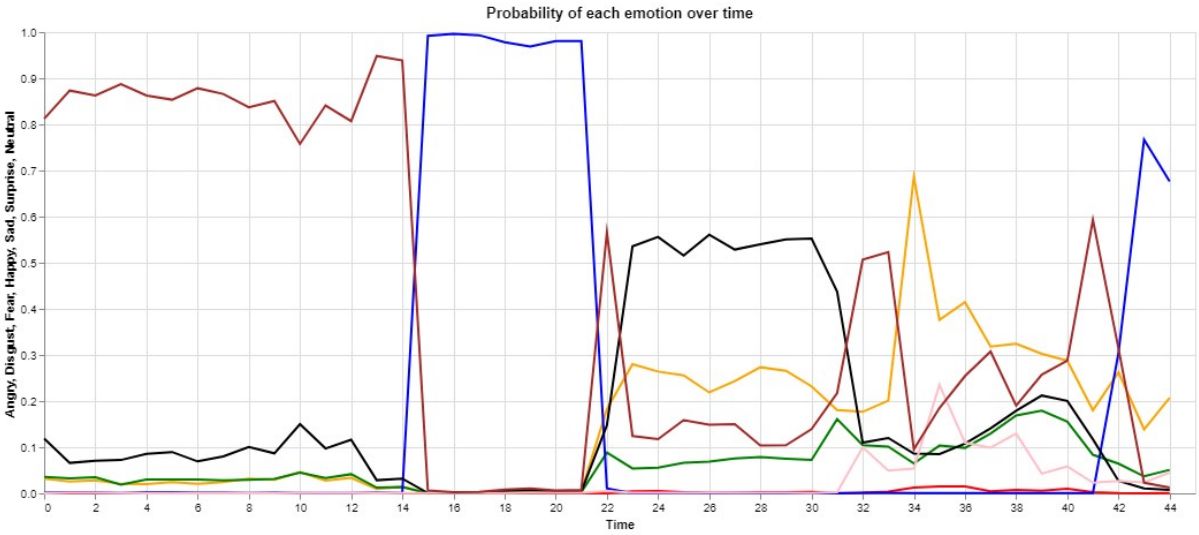
Neutral

- Angry : 2%
- Disgust : 0%
- Fear : 1%
- Happy : 23%
- Neutral : 22%
- Sad : 5%
- Surprise : 43%

Over time



Back


Probability of each emotion over time
Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral
Time
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44

References

1. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), https://zenodo.org/record/1188976/?f=3.XAcEs5NKhQK

2. The Facial Emotion Recognition Challenge from Kaggle, https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

3. End-to-End Multimodal Emotion Recognition using Deep Neural Networks,

https://arxiv.org/pdf/1704.08619.pdf

4. Facial Expression Recognition using Convolutional Neural Networks: State of the Art, https://arxiv.org/pdf/1612.02903.pdf

5. B.Basharirad, and M.Moradhaseli. Speech emotion recognition methods: A literature review. AIP Conference Proceedings 2017. https://aip.scitation.org/doi/pdf/10.1063/1.5005438

6. L.Chen, M.Mao, Y.Xue and L.L.Cheng. Speech emotion recognition: Features and classication models. Digit. Signal Process, vol 22 Dec. 2012.

7. T.Vogt, E.Andre and J.Wagner. Automatic Recognition of Emotions from Speech: A Review of the Literature and Recommendations for Practical Realisation. Aectand Emotion in Human-Computer Interaction, 2008.

8. T.Vogt and E.Andre. Improving Automatic Emotion Recognition from Speech via Gender Dierentiation. Language Resources and Evaluation Conference, 2006.

9. T.Giannakopoulos. pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis. Dec. 2015https://doi.org/10.1371/journal.pone.0144610

The End