Alexandria University- Faculty of Engineering

Computer and Systems Engineering Department

Pattern Recognition

Lab 3

Speech Emotion Recognition

Name	ID	Email
Ahmed Gamal Mahmoud	18010083	ahmed.gamal5551.ag@gmail.com
Marwan Mohamed Saad	18011736	marwangabal99@gmail.com
Shehab Mohamed Saad	18010857	shehab.mohamed2104@gmail.com

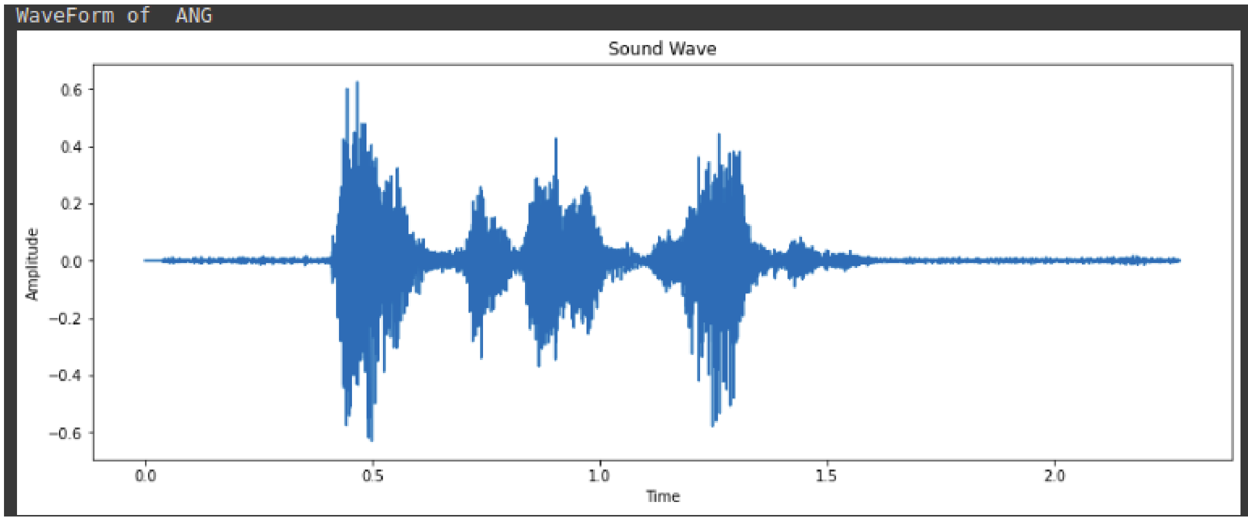
Colab link

Downloading dataset and understanding the format:

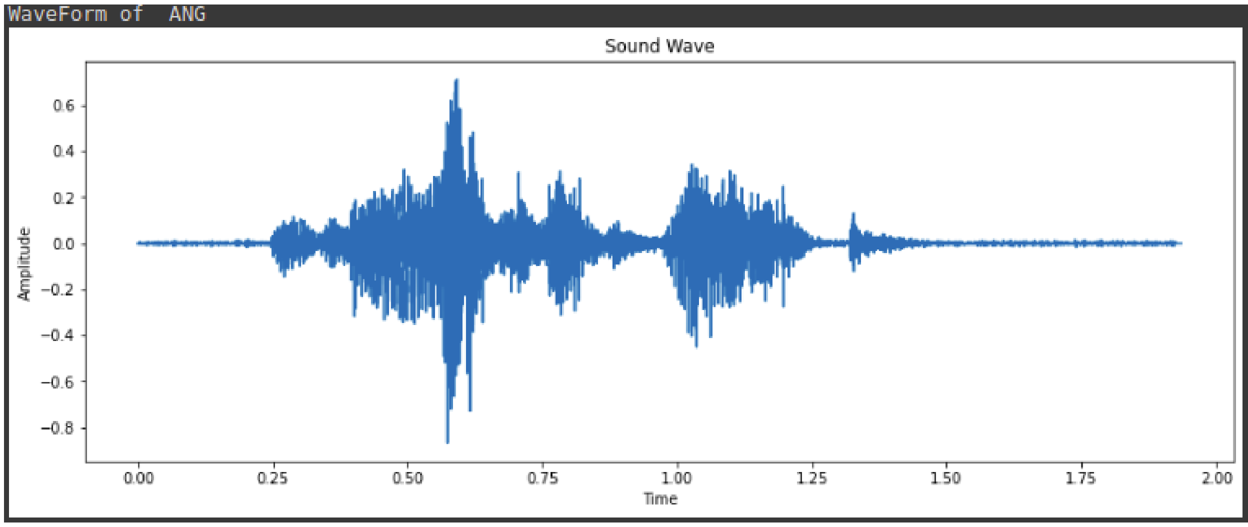
- We uploaded the dataset from kaggle directly to our colab and to do so, we needed to store kaggle.json which contain a key in a drive
- Size of the CREMA dataset is 7442 audios, each with different words and emotions..
- We used librosa library to load our audios as a time series ndarray while also storing each audio with its specific emotion.
- In loading the audio with librosa, We used sr=None as a parameter in librosa.load instead of the default sr Because sr=None preserves the native sampling rate of the audio file
- We plotted the time series audio waveform with it's sampling rate with time at x-axis and audios reading at y-axis
- The plotting functions takes the index of the specified audio to be plotted
- We made a function to play a specified audio and to listen to it using IPython.display.Audio

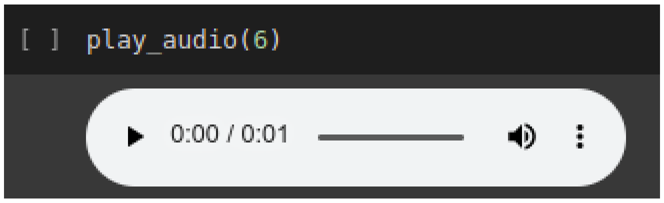
  Functions:

  - read_audio_librosa(directory)
  - plot_waveform_librosa(pos)
  - play_audio(pos)

- Waveform of first audio in CREMA
  - 

- Wave form of the 6th audio

- ○ 

- Audio playing for 6th audio
    - ○ 
```
[ ] play_audio(6)
```

Create the Feature space:

- In 1D :
    - We had multiple features
        - Zero Crossing rate : which get the value for a time series audio
        - Energy : Calculate root mean square value for each frame
        - Chroma Cens : Computes the Chroma Energy Normalized
        - Spectral Centroid : Calculate the centroid of a frame after normalizing the frame and treated as distribution over frequency bins
        - Spectral Bandwidth : Compute p'th order spectral bandwidth where p is the power to raise deviation from spectral centroid
        - Spectral rolloff : Compute rolloff frequency which is for each frame as the center frequency for a spectrogram bin
    - Functions:
        - zeroCross(D);
        - energyValue(D)
        - chroma_energy_norm(D)
        - spectral_centroid(D)

        - spectral_bandwidth(D)
        - spectral_rolloff(D)
- In 2D:
    - We converted the audio waveform into melspectrogram with librosa
        - MelSpectrogram : Take a time series as input and computer it’s magnitude spectrogram and then mapped onto the mel scale
    - Functions:
        - convertToSpectrogram(D,rate)
- In the 1D functions like (zero crossing range and energy), We had parameters like frame length which value is computed in that range and another parameter like hop length which is the number of samples to advance in each frame. We looked up and each combination of these parameters results in different features.Therefore, We stick to the default values.
- In the 2D function (melspectrogram) we entered the rate as a parameter with our time series ndarray because we modified the rate to our native sampling rate (16000) instead of the default rate (22050)

Data Augmentation:

- In 1D model we used 2 types of data augmentation
    - Noise aug. (adding noise to audio)
    - Stretch aug. (fasten or slown the audio)

    We do this because of overfitting and to get high accuracy
- In 2D model we used 1 types of data augmentation
    - Noise aug. (adding noise to audio)

    We do this because of overfitting and to get high accuracy

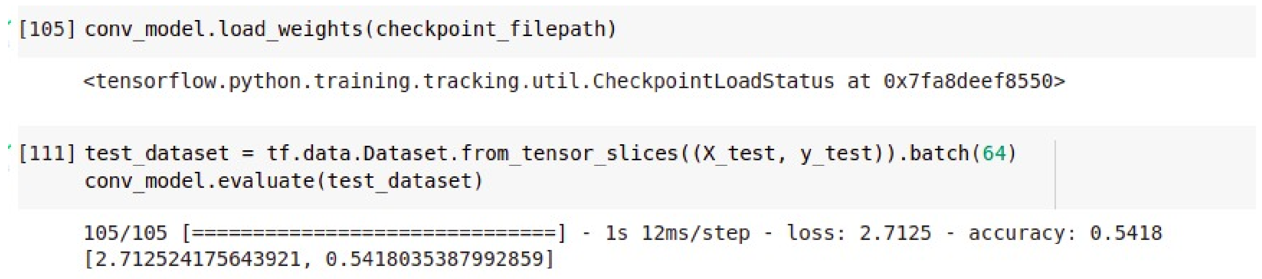
    We can’t use 2 types because of ram capacity

Building the model:

Designing models:

- We designed multiple models and after some testing we reached that the model used gives the best results
- Our model (1D) consists of 9 layers:
    - 1st layer contains : 2 conv1D(n = 32, f =3), 1 max pool layer (pool size = 2)
    - 2nd layer contains: 2 conv1D(n = 64,f=3), 1 max pool layer (pool size = 2)
    - 3rd layer contains: 2 conv1D(n = 128,,f=3), 1 max pool layer (pool size = 2)
    - 4th layer contains : 1 conv1D(n = 256,,f=3), 1 max pool layer (pool size = 5)
    - 5th layer contains : 1 conv1D(512), 1 max pool layer (pool size = 5)
    - 6th layer contains: flatten previous layers
    - 7th layer contains: dense layer (256)
    - 8th layer contains: dropout with percentage(0.25)
    - 9th layer contains: soft max

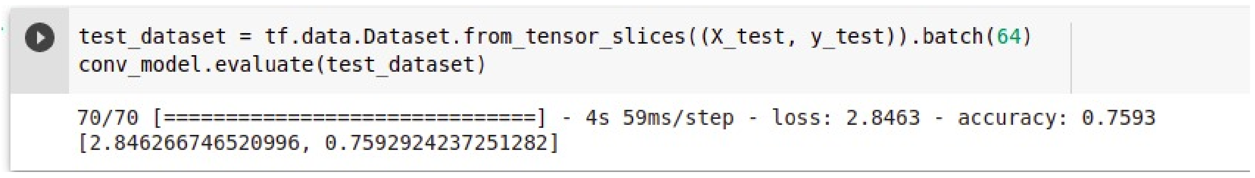
after using best weights of val_acc of 1D Model:



```
[105] conv_model.load_weights(checkpoint_filepath)

      <tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fa8deef8550>

[111] test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test)).batch(64)
      conv_model.evaluate(test_dataset)

      105/105 [==============================] - 1s 12ms/step - loss: 2.7125 - accuracy: 0.5418
      [2.712524175643921, 0.5418035387992859]
```

- Our model (2D) consists of 9 layers:
    - 1st layer contains : 2 conv2D(n = 32, f =3), 1 max pool layer (pool size = 2)
    - 2nd layer contains: 2 conv2D(n = 64,f=3), 1 max pool layer (pool size = 2)
    - 3rd layer contains: 1 conv2D(n = 128,f=3), 1 max pool layer (pool size = 2)
    - 4th layer contains : 1 conv2D(n = 256,f=3), 1 max pool layer (pool size = 5)
    - 5th layer contains: flatten previous layers
    - 6th layer contains: dense layer (256)
    - 7th layer contains: dense layer (128)
    - 8th layer contains: dropout with percentage(0.5)
    - 9th layer contains: soft max

after using best weights of val_acc of 2D Model:



```
test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test)).batch(64)
conv_model.evaluate(test_dataset)

70/70 [==============================] - 4s 59ms/step - loss: 2.8463 - accuracy: 0.7593
[2.846266746520996, 0.7592924237251282]
```

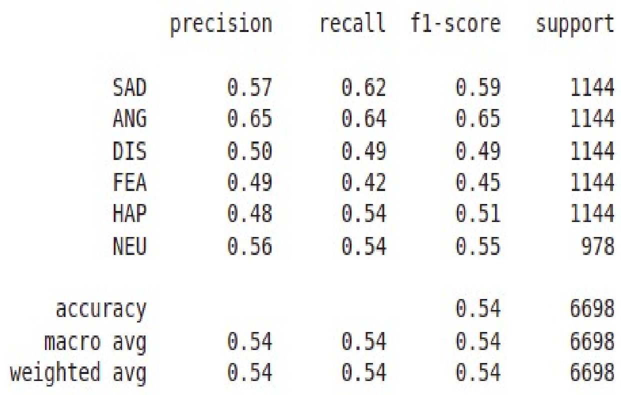
Big Picture:

These results based on best validation accuracy which saved in 50s epochs

But the model still running until epoch = 100

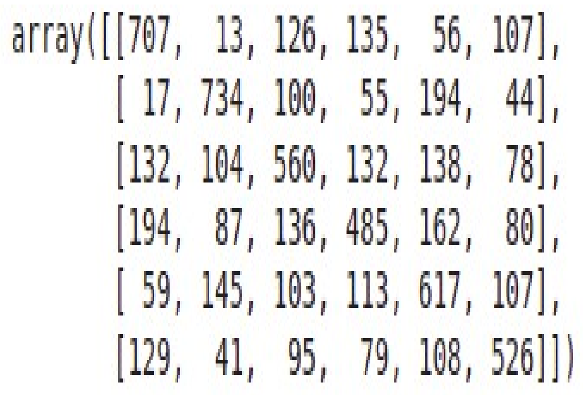
1D Model:

Classification Report results (accuracy & F1 score):

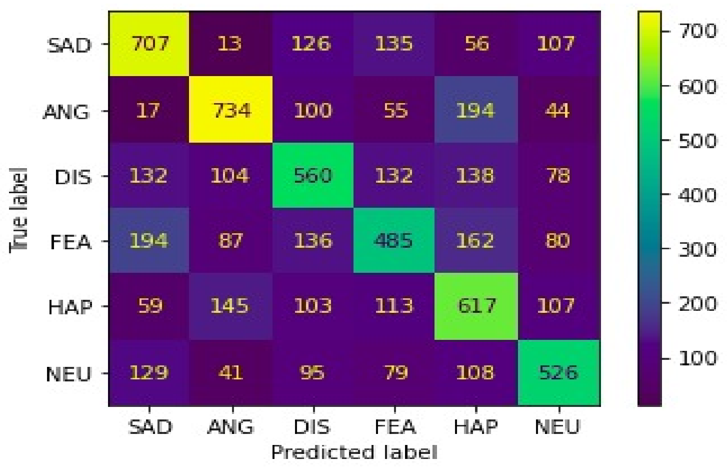


```
              precision    recall  f1-score   support

         SAD       0.57      0.62      0.59      1144
         ANG       0.65      0.64      0.65      1144
         DIS       0.50      0.49      0.49      1144
         FEA       0.49      0.42      0.45      1144
         HAP       0.48      0.54      0.51      1144
         NEU       0.56      0.54      0.55       978

    accuracy                           0.54      6698
   macro avg       0.54      0.54      0.54      6698
weighted avg       0.54      0.54      0.54      6698
```

Confusion Matrix:

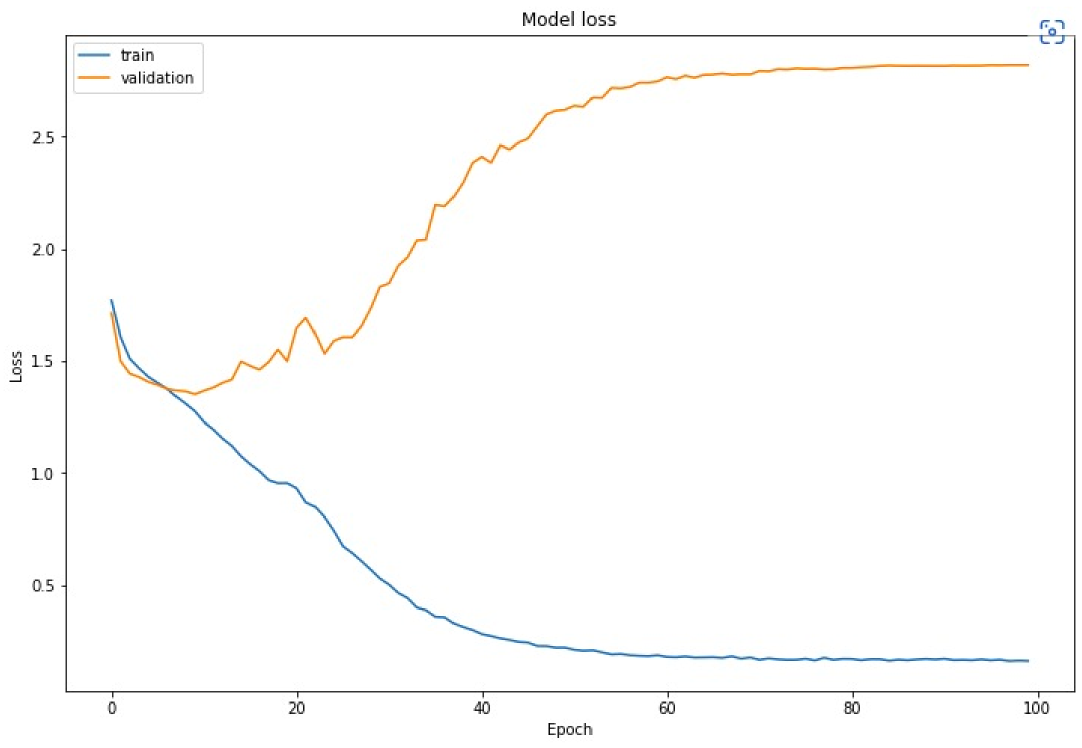


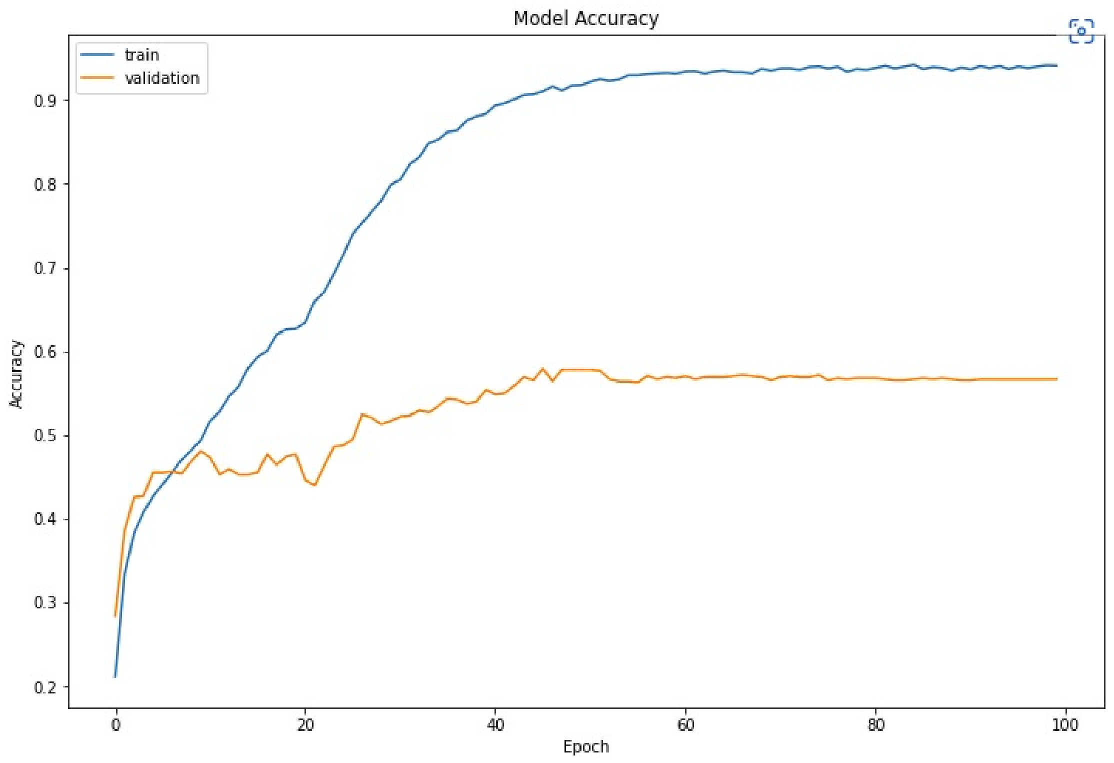
```
array([[707,  13, 126, 135,  56, 107],
       [ 17, 734, 100,  55, 194,  44],
       [132, 104, 560, 132, 138,  78],
       [194,  87, 136, 485, 162,  80],
       [ 59, 145, 103, 113, 617, 107],
       [129,  41,  95,  79, 108, 526]])
```



The most confusing classes are : (FEA, SAD), (HAP, ANG), (HAP, FEA), (DIS, FEA)

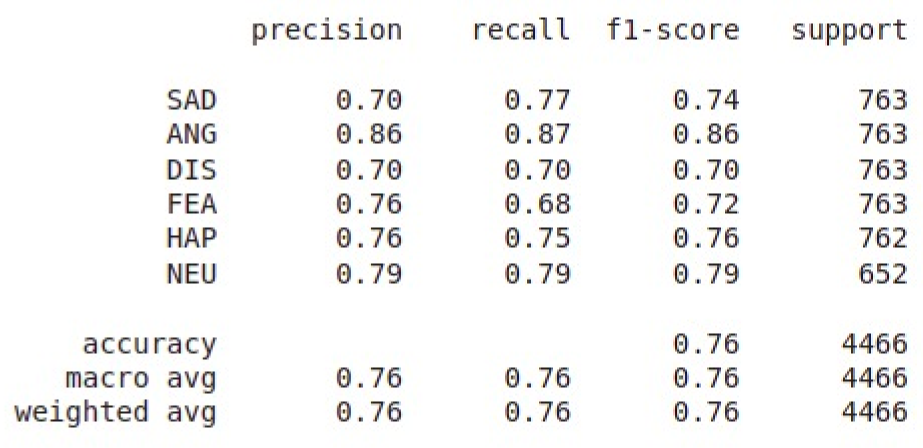
Accuracy & Loss graph:



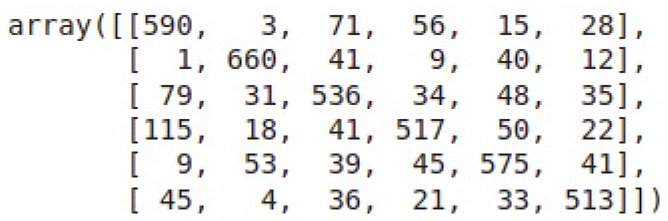


2D Model:

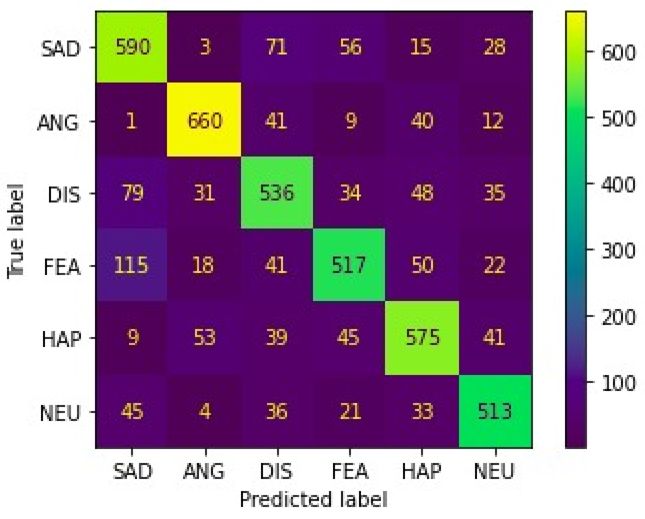
Classification Report results (accuracy & F1 score):


```
              precision    recall  f1-score   support

         SAD       0.70      0.77      0.74       763
         ANG       0.86      0.87      0.86       763
         DIS       0.70      0.70      0.70       763
         FEA       0.76      0.68      0.72       763
         HAP       0.76      0.75      0.76       762
         NEU       0.79      0.79      0.79       652

    accuracy                           0.76      4466
   macro avg       0.76      0.76      0.76      4466
weighted avg       0.76      0.76      0.76      4466
```

Confusion Matrix:



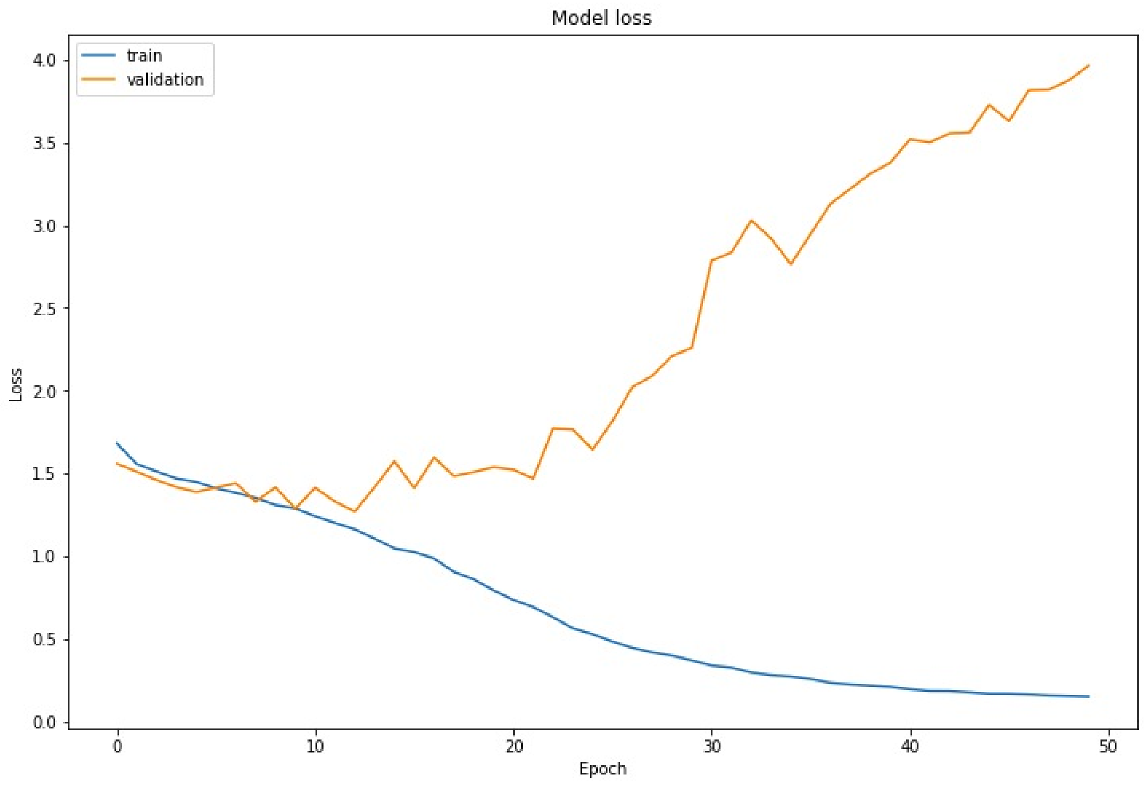
```
array([[590,   3,  71,  56,  15,  28],
       [  1, 660,  41,   9,  40,  12],
       [ 79,  31, 536,  34,  48,  35],
       [115,  18,  41, 517,  50,  22],
       [  9,  53,  39,  45, 575,  41],
       [ 45,   4,  36,  21,  33, 513]])
```

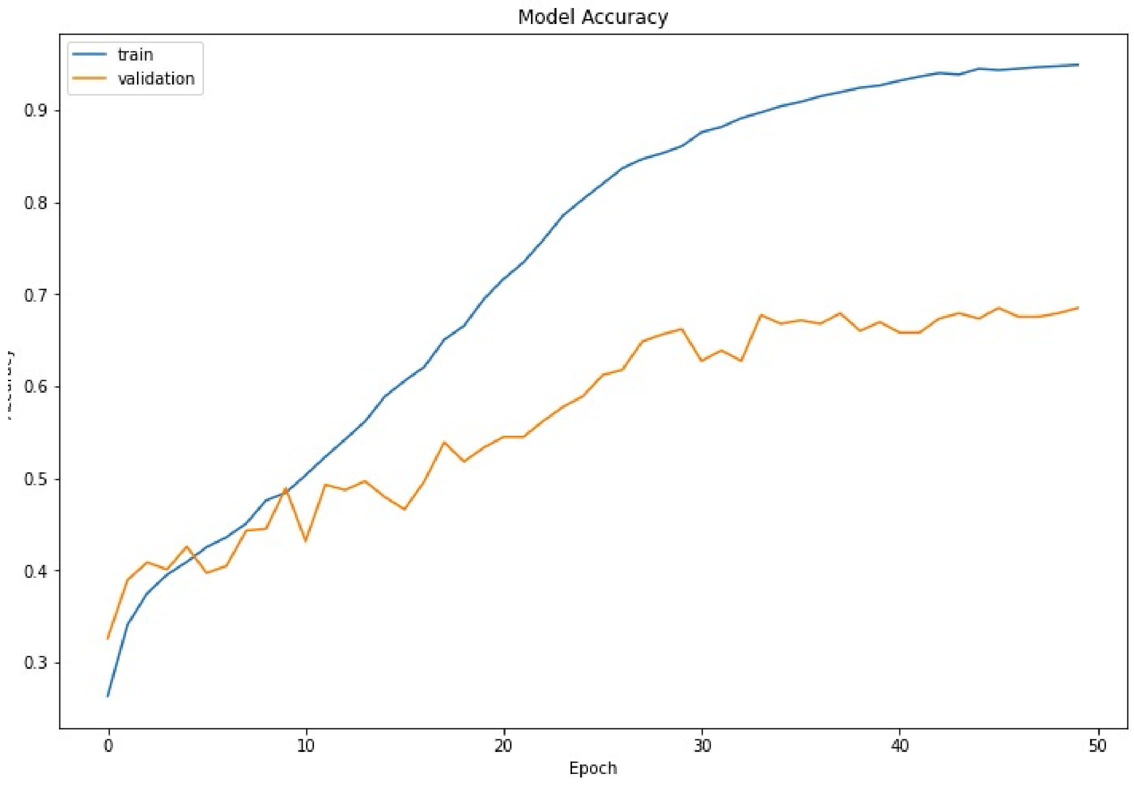


	SAD	ANG	DIS	FEA	HAP	NEU
SAD	590	3	71	56	15	28
ANG	1	660	41	9	40	12
DIS	79	31	536	34	48	35
FEA	115	18	41	517	50	22
HAP	9	53	39	45	575	41
NEU	45	4	36	21	33	513

The most confusing classes are : (FEA, SAD)

Accuracy & Loss graph:


Model loss
train
validation
Loss
Epoch


Model Accuracy
train
validation
Epoch