Pattern Recognition – Spring 2023

Speech Emotion Recognition

AHMED DUSUKI	6856
MANAR AMGAD	7113
NADA ELWAZANE	6876

1D CNN

Procedure:

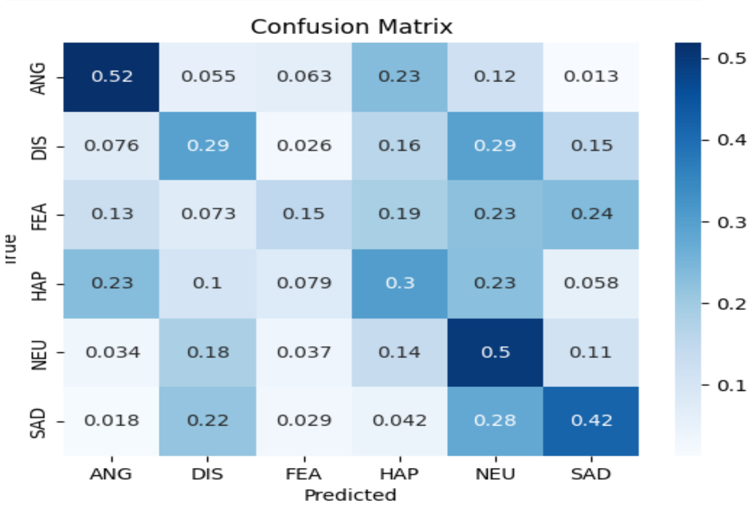
1- Feature Selection:

We used librosa library for feature extraction due to it's ease of use and many extraction features available.

Due to the large number of features available we started to apply incremental feature selection technique ,keeping ZCR ,energy in every attempt as it's mentioned in the assignment pdf .

Then , we added MFCC (n=20) due to it's importance and it was implemented in most papers making us confident using it .

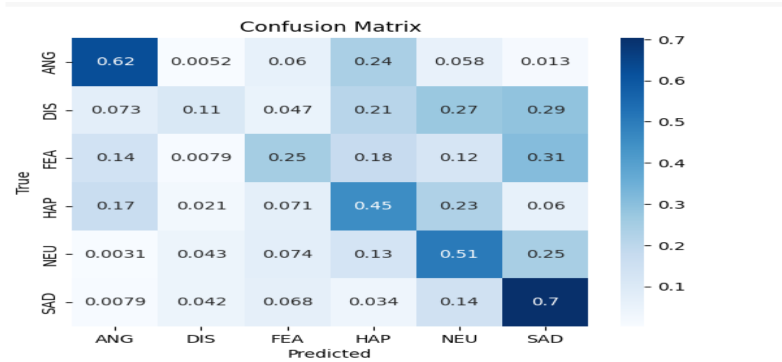
```
140/140 [==============================] - 1s 5ms/step
              precision    recall  f1-score   support

           0       0.52      0.52      0.52       382
           1       0.33      0.29      0.31       381
           2       0.40      0.15      0.22       381
           3       0.29      0.30      0.30       382
           4       0.27      0.50      0.35       326
           5       0.43      0.42      0.42       381

    accuracy                           0.36      2233
   macro avg       0.37      0.36      0.35      2233
weighted avg       0.38      0.36      0.35      2233
```



Looking at the results they are promising but we thought we could do better so ,we added chroma , The use of chroma degraded the disgust classification accuracy but overall improved the model.

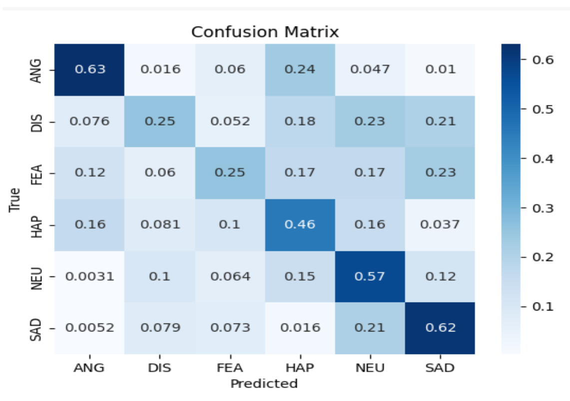
140/140 [==============================] - 2s 16ms/step

	precision	recall	f1-score	support
0	0.61	0.62	0.62	382
1	0.49	0.11	0.18	381
2	0.45	0.25	0.32	381
3	0.37	0.45	0.41	382
4	0.34	0.51	0.41	326
5	0.44	0.70	0.54	381
accuracy			0.44	2233
macro avg	0.45	0.44	0.41	2233
weighted avg	0.45	0.44	0.41	2233



Then we added Roll off feature ,

The addition of spectral roll off improved disgust by almost double.



140/140 [==============================] - 2s 15ms/step

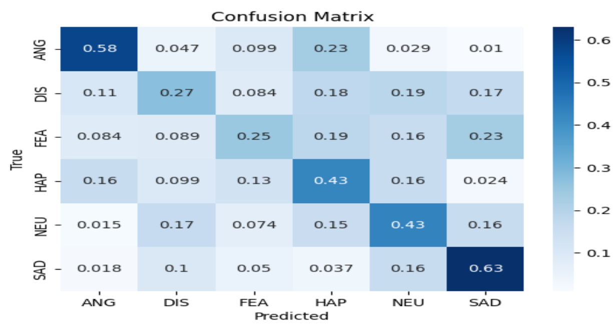
	precision	recall	f1-score	support
0	0.63	0.63	0.63	382
1	0.44	0.25	0.32	381
2	0.42	0.25	0.32	381
3	0.39	0.46	0.42	382
4	0.37	0.57	0.45	326
5	0.51	0.62	0.56	381
accuracy			0.46	2233
macro avg	0.46	0.46	0.45	2233
weighted avg	0.46	0.46	0.45	2233

Then we added LDA,

due to most papers recommending dimensionality reduction

LDA improved the overall accuracies of most classes.

```
140/140 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

           0       0.60      0.58      0.59       382
           1       0.36      0.27      0.31       381
           2       0.37      0.25      0.30       381
           3       0.36      0.43      0.39       382
           4       0.35      0.43      0.38       326
           5       0.53      0.63      0.57       381

    accuracy                           0.43      2233
   macro avg       0.43      0.43      0.42      2233
weighted avg       0.43      0.43      0.43      2233
```



Then we added Tonnetz ,

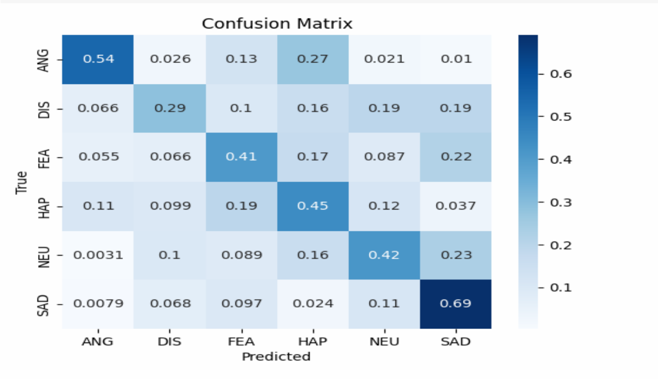
Happy and disgust tied for second to last place only due to tonnetz improving the disgust classification accuracy


```
140/140 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.63      0.52      0.57       382
           1       0.35      0.35      0.35       381
           2       0.37      0.28      0.32       381
           3       0.36      0.37      0.36       382
           4       0.37      0.43      0.40       326
           5       0.53      0.65      0.59       381

    accuracy                           0.43      2233
   macro avg       0.43      0.43      0.43      2233
weighted avg       0.44      0.43      0.43      2233
```

Then we switch to MFCC but with n =100,

because we thought we could benefit more from it

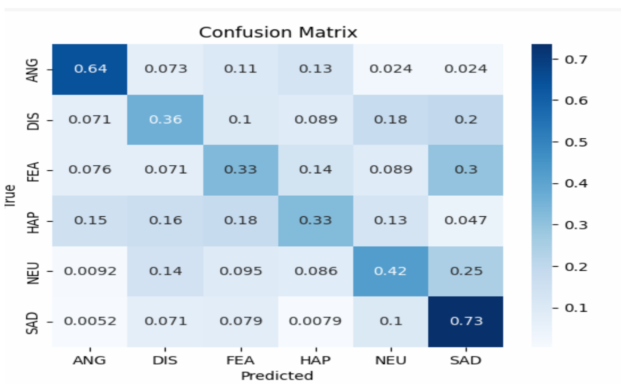
The use of mfcc n=100 lead to an overall increase in per class accuracy.




```
140/140 [==============================] - 1s 2ms/step
              precision    recall  f1-score   support

           0       0.69      0.54      0.61       382
           1       0.45      0.29      0.35       381
           2       0.41      0.41      0.41       381
           3       0.37      0.45      0.40       382
           4       0.40      0.42      0.41       326
           5       0.51      0.69      0.59       381

    accuracy                           0.47      2233
   macro avg       0.47      0.46      0.46      2233
weighted avg       0.47      0.47      0.46      2233
```

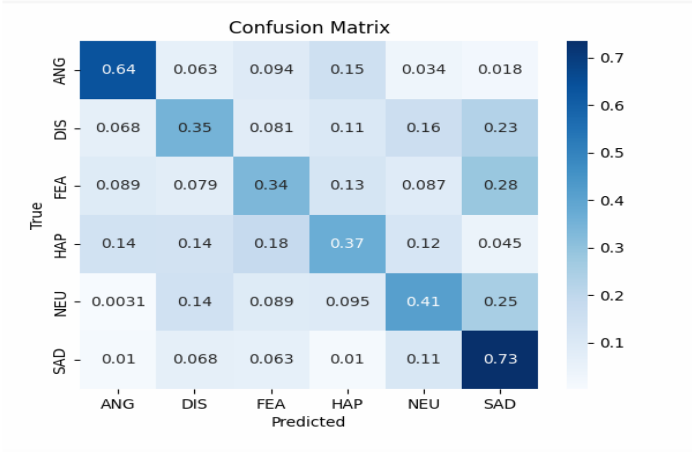
Then we added spectral contrast ,

The addition of spectral contrast greatly improved Disgust's accuracy pulling it from the bottom rank.




```
140/140 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.67      0.64      0.66       382
           1       0.42      0.36      0.39       381
           2       0.38      0.33      0.35       381
           3       0.43      0.33      0.37       382
           4       0.41      0.42      0.42       326
           5       0.49      0.73      0.59       381

    accuracy                           0.47      2233
   macro avg       0.47      0.47      0.46      2233
weighted avg       0.47      0.47      0.46      2233
```

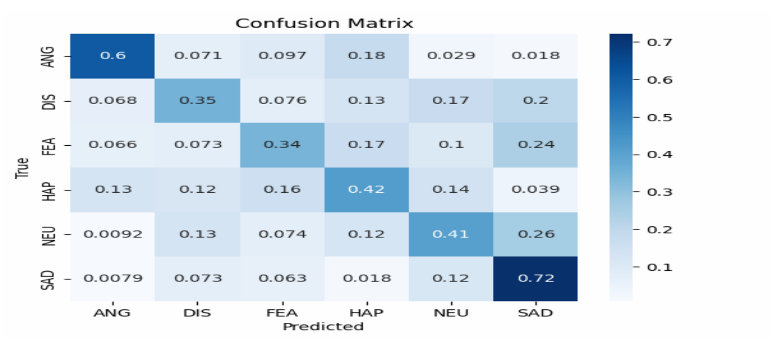
Then we added spectral flatness , Happiness was slightly improved.



140/140 [==============================] - 0s 2ms/step

	precision	recall	f1-score	support
0	0.67	0.64	0.65	382
1	0.43	0.35	0.39	381
2	0.40	0.34	0.37	381
3	0.44	0.37	0.40	382
4	0.41	0.41	0.41	326
5	0.48	0.73	0.58	381
accuracy			0.48	2233
macro avg	0.47	0.47	0.47	2233
weighted avg	0.47	0.48	0.47	2233

We added spectral bandwidth , Sadness is greatly improved by the use of spectral bandwidth.

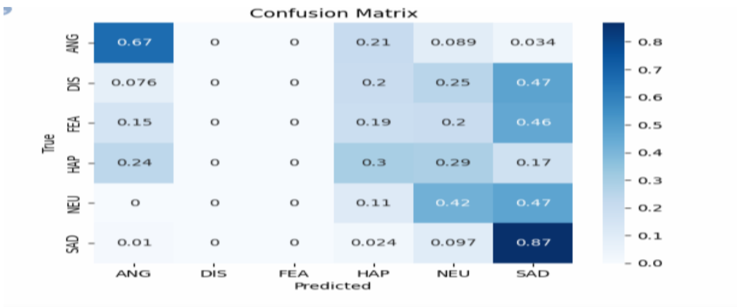


140/140 [==============================] - 0s 2ms/step

	precision	recall	f1-score	support
0	0.69	0.60	0.64	382
1	0.44	0.35	0.39	381
2	0.43	0.34	0.38	381
3	0.41	0.42	0.41	382
4	0.38	0.41	0.39	326
5	0.50	0.72	0.59	381
accuracy			0.48	2233
macro avg	0.47	0.47	0.47	2233
weighted avg	0.48	0.48	0.47	2233

Then , we decided to try out PCA with n =10 for features reduction instead of LDA ,

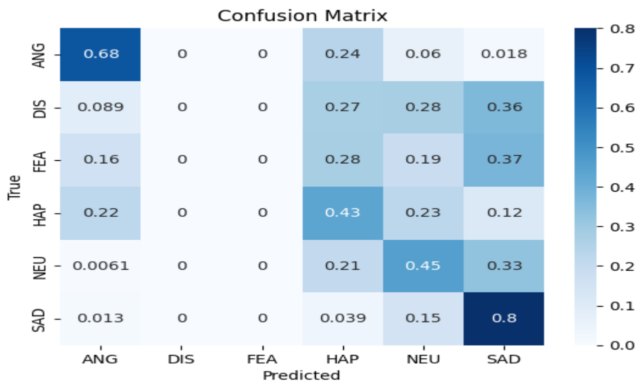
Using PCA (n=10) with our model leads to it not classifying any of the testing samples as disgust or fear. Changing the number of components may be helpful.




```
140/140 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

           0       0.58      0.67      0.62       382
           1       0.00      0.00      0.00       381
           2       0.00      0.00      0.00       381
           3       0.29      0.30      0.30       382
           4       0.28      0.42      0.34       326
           5       0.36      0.87      0.51       381

    accuracy                           0.38      2233
   macro avg       0.25      0.38      0.29      2233
weighted avg       0.25      0.38      0.29      2233
```

Then ,We tried out PCA with n=20,

Using PCA (n=20) with our model leads to it not classifying any of the testing samples as disgust or fear. Changing the number of components did not help.

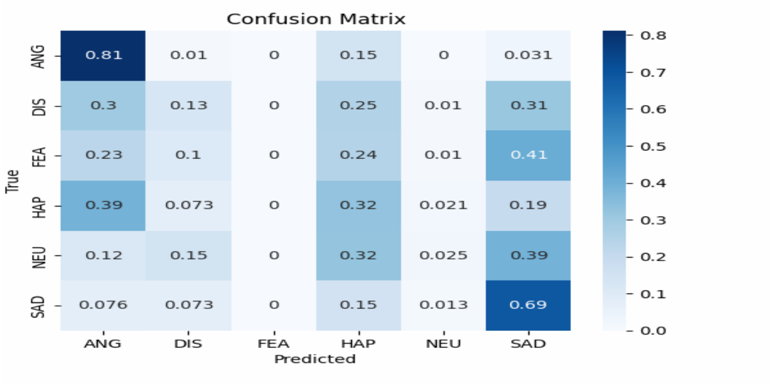



```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.58      0.68      0.63       382
           1       0.00      0.00      0.00       381
           2       0.00      0.00      0.00       381
           3       0.30      0.43      0.36       382
           4       0.30      0.45      0.36       326
           5       0.41      0.80      0.54       381

    accuracy                           0.39      2233
   macro avg       0.27      0.40      0.31      2233
weighted avg       0.27      0.39      0.31      2233
```

2- We decided to change the Model’s architecture using previous features but without LDA

This model does not predict any testing data as fear and very few as neutral making them the most confusing classes.

Switching from SGD to Adam optimizer may be beneficial.

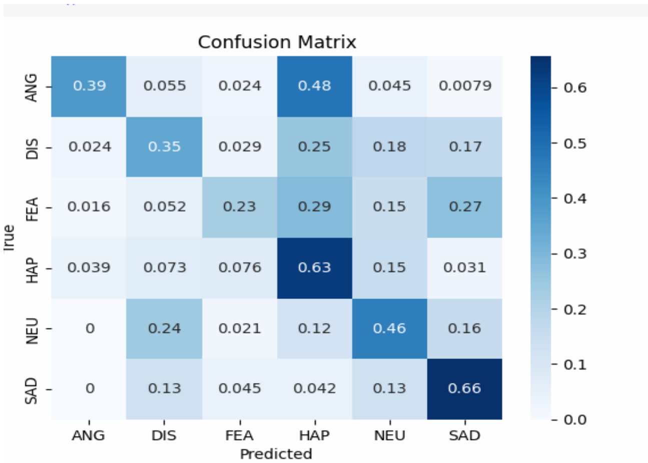


140/140 [==============================] - 1s 5ms/step

	precision	recall	f1-score	support
0	0.43	0.81	0.56	382
1	0.25	0.13	0.17	381
2	0.00	0.00	0.00	381
3	0.23	0.32	0.27	382
4	0.28	0.02	0.05	326
5	0.35	0.69	0.46	381
accuracy			0.34	2233
macro avg	0.26	0.33	0.25	2233
weighted avg	0.26	0.34	0.26	2233

Then we switched to Adam opt and with slower rate for LR:

Using Adam optimizer and a slightly lower rate for LR reduction disgust and fear are being predicted again though they still remain the most confusing to classify.



140/140 [==============================] - 1s 5ms/step

	precision	recall	f1-score	support
0	0.83	0.39	0.53	382
1	0.40	0.35	0.37	381
2	0.54	0.23	0.32	381
3	0.35	0.63	0.45	382
4	0.38	0.46	0.41	326
5	0.52	0.66	0.58	381
accuracy			0.45	2233
macro avg	0.50	0.45	0.44	2233
weighted avg	0.51	0.45	0.44	2233

Using LDA:

```python
def create_model4(input_size):
  model = Sequential()
  model.add(Conv1D(256, 3, padding='same',input_shape=(input_size,1))) #1
  model.add(Activation('relu'))
  model.add(Conv1D(256, 3, padding='same')) #2
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dropout(0.25))
  model.add(MaxPooling1D(pool_size=(3)))
  model.add(Conv1D(128, 3, padding='same')) #3
  model.add(Activation('relu'))
  model.add(Conv1D(128, 3, padding='same')) #4
  model.add(Activation('relu'))
  model.add(Conv1D(128, 3, padding='same')) #5
  model.add(Activation('relu'))
  model.add(Conv1D(128, 3, padding='same')) #6
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dropout(0.25))
 # max pool
  model.add(MaxPool1D( pool_size=3,padding='same'))
  model.add(Conv1D(64, 3, padding='same')) #7
  model.add(Activation('relu'))
  model.add(Conv1D(64, 3, padding='same')) #8
  model.add(Activation('relu'))
  model.add(Flatten())
  model.add(Dense(6)) #9
  model.add(Activation('softmax'))
  opt = keras.optimizers.SGD(lr=0.001, momentum=0.0, decay=0.0, nesterov=False)
  model.compile(loss = 'categorical_crossentropy',optimizer =opt,metrics = ['accuracy'])
  return model
```

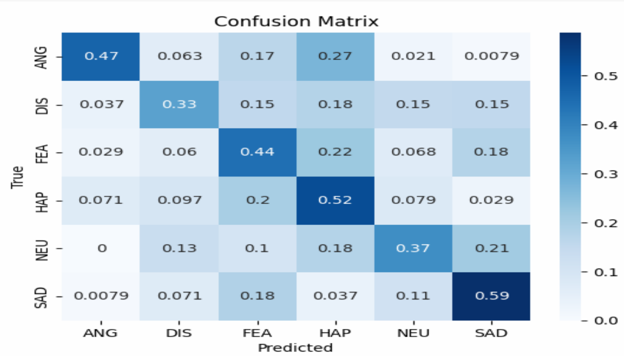
```
140/140 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

           0       0.69      0.58      0.63       382
           1       0.43      0.38      0.40       381
           2       0.41      0.37      0.39       381
           3       0.39      0.42      0.41       382
           4       0.40      0.50      0.44       326
           5       0.56      0.62      0.59       381

    accuracy                           0.48      2233
   macro avg       0.48      0.48      0.48      2233
weighted avg       0.48      0.48      0.48      2233
```

3- We decided to change the Model's architecture and go deeper using previous features and LDA

We started with input layer 512 and then went the way down to 32 and we used dense layer with size 256

```
def create_model3(input_size):
  model=Sequential()
  model.add(Conv1D(512, kernel_size=3, strides=1, padding='same', activation='relu', input_shape=(input_size, 1)))
  model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))

  model.add(Conv1D(256, kernel_size=3, strides=1, padding='same', activation='relu'))
  model.add(BatchNormalization())

  model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
  model.add(Conv1D(128, kernel_size=3, strides=1, padding='same', activation='relu'))
  model.add(Conv1D(64, kernel_size=3, strides=1, padding='same', activation='relu'))
  model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
  model.add(Conv1D(32, kernel_size=3, strides=1, padding='same', activation='relu'))

  model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
  model.add(Dropout(0.25))

  model.add(Flatten())
  model.add(Dense(256, activation='relu'))
  model.add(Dropout(0.25))

  model.add(Dense(6, activation='softmax'))
  opt=tensorflow.keras.optimizers.Adam(
    learning_rate=0.001
)

  model.compile(loss = 'categorical_crossentropy',optimizer = 'Adam',metrics = ['accuracy'])
  return model
```

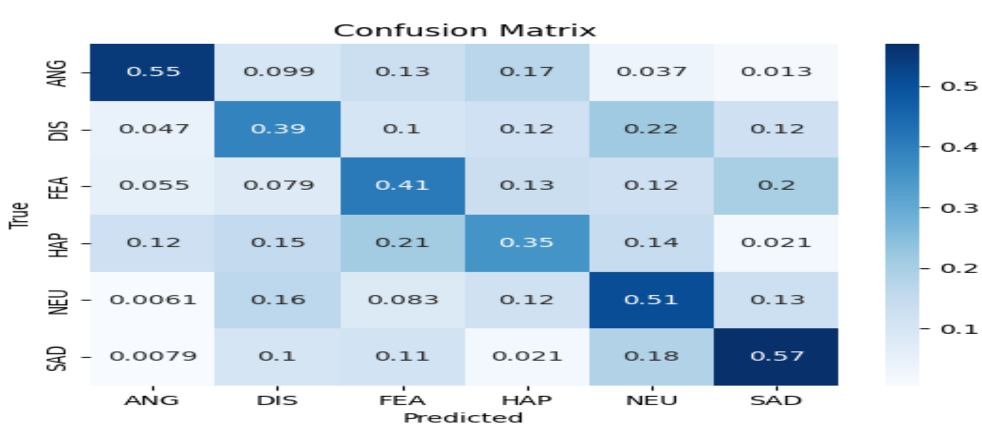
```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.76      0.47      0.58       382
           1       0.44      0.33      0.38       381
           2       0.36      0.44      0.40       381
           3       0.38      0.52      0.44       382
           4       0.43      0.37      0.40       326
           5       0.52      0.59      0.55       381

    accuracy                           0.46      2233
   macro avg       0.48      0.45      0.46      2233
weighted avg       0.48      0.46      0.46      2233
```



4-

```
[ ] def create_model5(input_size):

      model=Sequential()
      model.add(Conv1D(512, kernel_size=3, strides=1, padding='same', activation='relu', input_shape=(input_size, 1)))
      model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))

      model.add(Conv1D(256, kernel_size=3, strides=1, padding='same', activation='relu'))
      model.add(BatchNormalization())

      model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
      model.add(Conv1D(128, kernel_size=3, strides=1, padding='same', activation='relu'))
      model.add(Conv1D(64, kernel_size=3, strides=1, padding='same', activation='relu'))
      model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
      model.add(Conv1D(32, kernel_size=3, strides=1, padding='same', activation='relu'))

      model.add(MaxPooling1D(pool_size=3, strides = 2, padding = 'same'))
      model.add(Dropout(0.25))

      model.add(Flatten())
      model.add(Dense(256, activation='relu'))
      model.add(Dropout(0.25))

      model.add(Dense(6, activation='softmax'))
      opt=tensorflow.keras.optimizers.Adam(
        learning_rate=0.001
    )
      model.compile(loss = 'categorical_crossentropy',optimizer ='Adam',metrics = ['accuracy'])


      return model
```

```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.70      0.55      0.61       382
           1       0.41      0.39      0.40       381
           2       0.39      0.41      0.40       381
           3       0.39      0.35      0.37       382
           4       0.38      0.51      0.43       326
           5       0.55      0.57      0.56       381

    accuracy                           0.46      2233
   macro avg       0.47      0.46      0.46      2233
weighted avg       0.47      0.46      0.46      2233
```



5-Cnn, Convolutional Neural Network. "Speech emotion recognition using convolutional neural network (CNN)." International Journal of Psychosocial Rehabilitation 24.8 (2020): 1-20

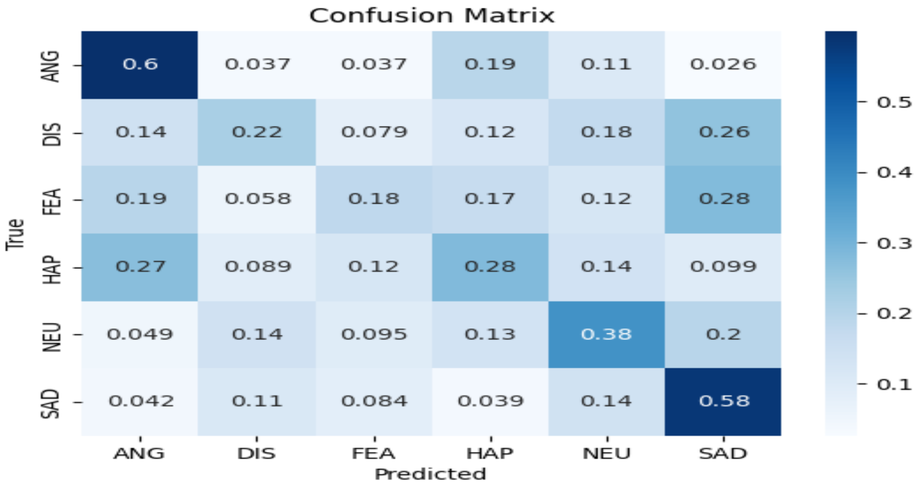
DOI:10.37200/IJPR/V24I8/PR280260

```python
def create_model():
  model = Sequential()
  model.add(Conv1D(128, 2,padding='same', input_shape=(5,1)))
  model.add(Activation('relu'))
  model.add(Conv1D(128, 2,padding='same'))
  model.add(Activation('relu'))
  model.add(Dropout(0.1))
  model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
  model.add(Conv1D(128, 2,padding='same'))
  model.add(Activation('relu'))
  model.add(Conv1D(128, 2,padding='same'))
  model.add(Activation('relu'))
  model.add(Conv1D(128, 2,padding='same'))
  model.add(Activation('relu'))
  model.add(Dropout(0.2))
  model.add(Conv1D(128, 2,padding='same'))
  model.add(Activation('relu'))
  model.add(Flatten())
  model.add(Dense(6))
  model.add(Activation('softmax'))
  opt=tensorflow.keras.optimizers.RMSprop(learning_rate=0.001)
  model.compile(loss = 'categorical_crossentropy',optimizer ='Adam',metrics = ['accuracy',get_f1])

  return model
```

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 216, 128)	768
activation_1 (Activation)	(None, 216, 128)	0
conv1d_2 (Conv1D)	(None, 216, 128)	82048
activation_2 (Activation)	(None, 216, 128)	0
dropout_1 (Dropout)	(None, 216, 128)	0
max_pooling1d_1 (MaxPooling1	(None, 27, 128)	0
conv1d_3 (Conv1D)	(None, 27, 128)	82048
activation_3 (Activation)	(None, 27, 128)	0
conv1d_4 (Conv1D)	(None, 27, 128)	82048
activation_4 (Activation)	(None, 27, 128)	0
conv1d_5 (Conv1D)	(None, 27, 128)	82048
activation_5 (Activation)	(None, 27, 128)	0
dropout_2 (Dropout)	(None, 27, 128)	0
conv1d_6 (Conv1D)	(None, 27, 128)	82048
activation_6 (Activation)	(None, 27, 128)	0
flatten_1 (Flatten)	(None, 3456)	0
dense_1 (Dense)	(None, 10)	34570
activation_7 (Activation)	(None, 10)	0

Total params: 445,578
Trainable params: 445,578
Non-trainable params: 0

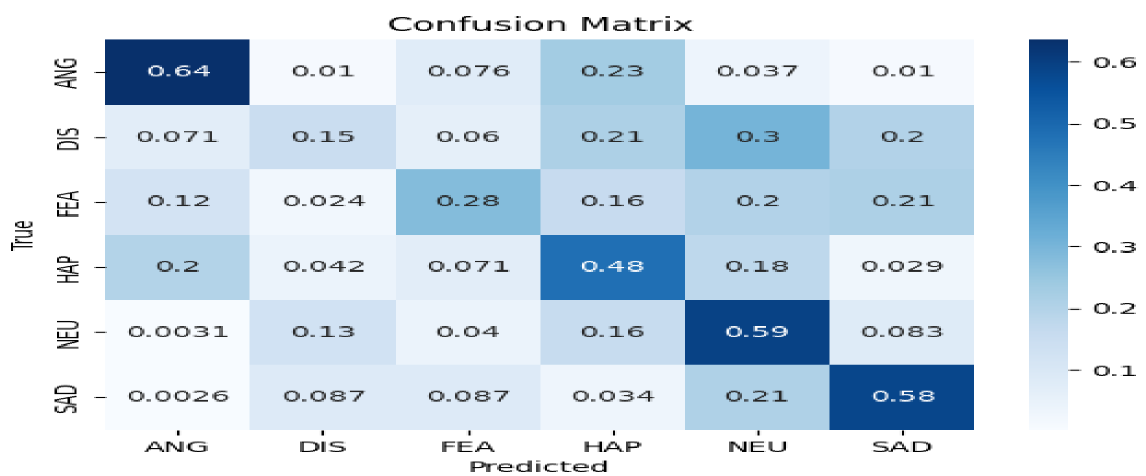
```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.46      0.60      0.52       382
           1       0.35      0.22      0.27       381
           2       0.31      0.18      0.23       381
           3       0.31      0.28      0.30       382
           4       0.33      0.38      0.35       326
           5       0.41      0.58      0.48       381

    accuracy                           0.38      2233
   macro avg       0.36      0.38      0.36      2233
weighted avg       0.36      0.38      0.36      2233
```



Disgust and Fear are the most confusing classes. The paper claimed 82% accuracy which was not achieved here. This may be due to the use of a different dataset. Learning rate was also not specified though we did experiment with different learning rates and settled on 0.001 as having the best accuracy but only marginally.

6-

```
[ ] def create_model7(input_size):
      #BUILD 1D CNN LAYERS
      model = tensorflow.keras.Sequential()
      model.add(layers.Conv1D(64, kernel_size=(10), activation='relu', input_shape=(input_size,1)))
      model.add(layers.Conv1D(128, kernel_size=(10),activation='relu',kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
      model.add(layers.MaxPooling1D(pool_size=(8)))
      model.add(layers.Dropout(0.4))
      model.add(layers.Conv1D(128, kernel_size=(10),activation='relu'))
      model.add(layers.MaxPooling1D(pool_size=(8)))
      model.add(layers.Dropout(0.4))
      model.add(layers.Flatten())
      model.add(layers.Dense(256, activation='relu'))
      model.add(layers.Dropout(0.4))
      model.add(layers.Dense(6, activation='sigmoid'))
      opt = keras.optimizers.Adam(lr=0.001)
      model.compile(loss='categorical_crossentropy', optimizer=opt,metrics=['accuracy'])
      return model
```



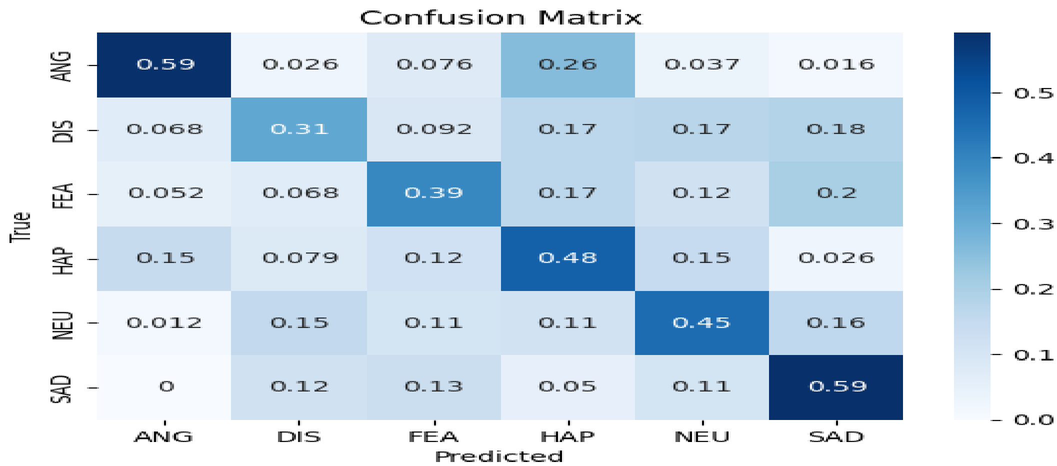

```
140/140 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

           0       0.62      0.64      0.63       382
           1       0.36      0.15      0.21       381
           2       0.46      0.28      0.35       381
           3       0.38      0.48      0.43       382
           4       0.35      0.59      0.44       326
           5       0.53      0.58      0.55       381

    accuracy                           0.45      2233
   macro avg       0.45      0.45      0.43      2233
weighted avg       0.45      0.45      0.43      2233
```

Most confusing classes are Disgust and Fear with 0.15 and 0.28 accuracy respectively.

Using Augmented training Data:

```
[ ] def create_model0():
        model=Sequential()
        model.add(Conv1D(512, kernel_size=5, strides=1, padding='same', activation='relu', input_shape=(258, 1)))
        model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

        model.add(Conv1D(256, kernel_size=5, strides=1, padding='same', activation='relu'))
        model.add(BatchNormalization())

        model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
        model.add(Conv1D(128, kernel_size=5, strides=1, padding='same', activation='relu'))
        model.add(Conv1D(64, kernel_size=5, strides=1, padding='same', activation='relu'))
        model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
        model.add(Conv1D(32, kernel_size=5, strides=1, padding='same', activation='relu'))

        model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
        model.add(Dropout(0.25))

        model.add(Flatten())
        model.add(Dense(256, activation='relu'))
        model.add(Dropout(0.25))

        model.add(Dense(6, activation='softmax'))
        opt=tensorflow.keras.optimizers.Adam(
          learning_rate=0.001
      )
        model.compile(loss = 'categorical_crossentropy',optimizer ='Adam',metrics = ['accuracy',get_f1])
        return model
```

```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.68      0.59      0.63       382
           1       0.43      0.31      0.36       381
           2       0.43      0.39      0.41       381
           3       0.39      0.48      0.43       382
           4       0.40      0.45      0.43       326
           5       0.51      0.59      0.55       381

    accuracy                           0.47      2233
   macro avg       0.47      0.47      0.47      2233
weighted avg       0.48      0.47      0.47      2233
```



Disgust and Fear remain the most confusing classes even after the use of augmentation. Augmentation did not improve the overall accuracy or f1-score.

Convolution 2D:

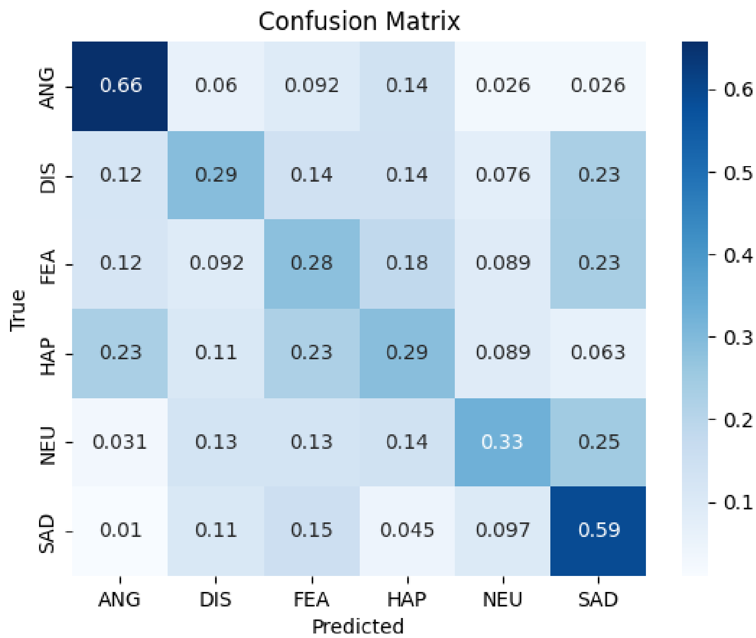
For convolution 2D, we used MelSpectrogram as our feature of choice due to its popularity and the fact that it was mentioned in the problem statement.

Our Initial Model Architecture was the following:

```
def create_model2d():
  input_shape=(30,216,1)
  CNNmodel = models.Sequential()
  CNNmodel.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
  CNNmodel.add(layers.MaxPooling2D((2, 2)))
  CNNmodel.add(layers.Dropout(0.2))
  CNNmodel.add(layers.Conv2D(64, (3, 3), activation='relu'))
  CNNmodel.add(layers.MaxPooling2D((2, 2)))
  CNNmodel.add(layers.Dropout(0.2))
  CNNmodel.add(layers.Conv2D(64, (3, 3), activation='relu'))
  CNNmodel.add(layers.Flatten())
  CNNmodel.add(layers.Dense(64, activation='relu'))
  CNNmodel.add(layers.Dropout(0.2))
  CNNmodel.add(layers.Dense(32, activation='relu'))
  CNNmodel.add(layers.Dense(6, activation='softmax'))

  CNNmodel.compile(loss = 'categorical_crossentropy',optimizer = 'Adam',metrics = ['accuracy',get_f1])
  return CNNmodel
```

Our testing evaluation results were as follows:

```
140/140 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.56      0.66      0.61       382
           1       0.38      0.29      0.33       381
           2       0.28      0.28      0.28       381
           3       0.32      0.29      0.30       382
           4       0.43      0.33      0.37       326
           5       0.44      0.59      0.50       381

    accuracy                           0.41      2233
   macro avg       0.40      0.41      0.40      2233
weighted avg       0.40      0.41      0.40      2233
```



The most confusing classes here are Fear 0.28,disgust 0.29 and happy 0.29 This model yields acc 0.41 and F1-score with weighted avg 0.40 and with macro average 0.40.

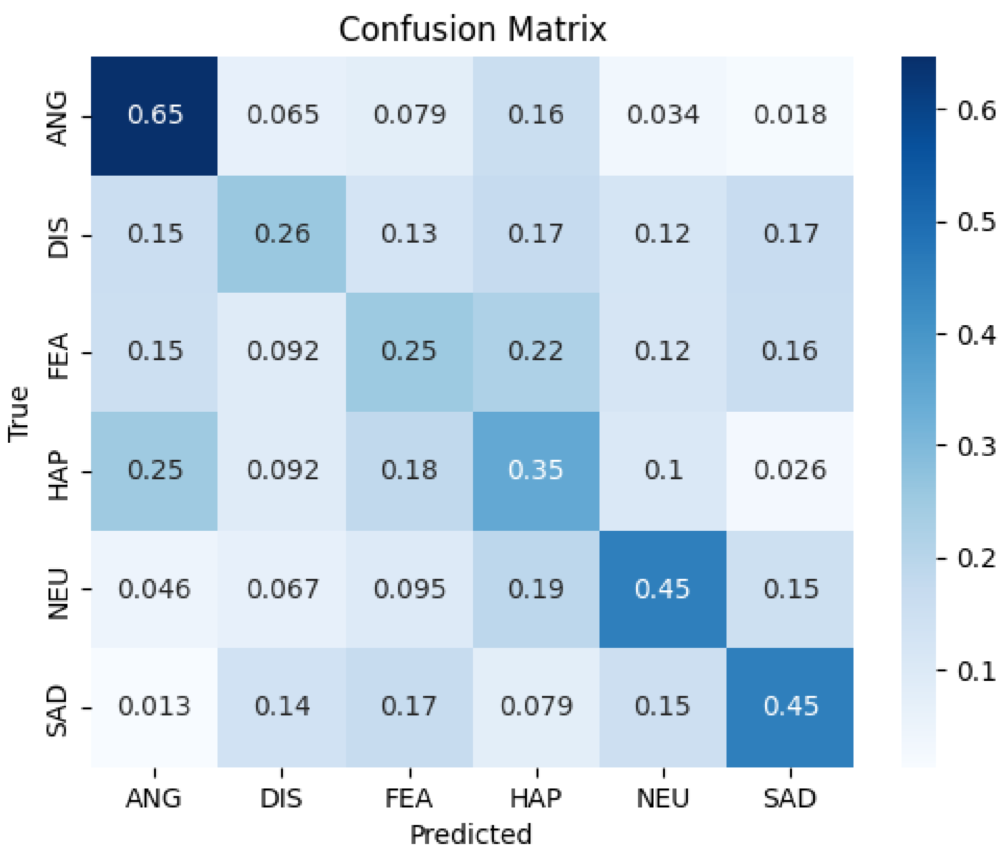
Learning rate for this model was the default 0.0010 we also used LR reduce on plateau with a rate of 0.9 to enable the model to take smaller learning steps to the optimal solution of the cost function.

We then attempted to start with a lower lr of 0.0001

Initial Model with lr=0.0001

This approach did not yield better results but is included to demonstrate our attempts at hyperparameter tuning.

```
140/140 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.52      0.65      0.57       382
           1       0.37      0.26      0.30       381
           2       0.28      0.25      0.27       381
           3       0.30      0.35      0.32       382
           4       0.42      0.45      0.44       326
           5       0.48      0.45      0.47       381

    accuracy                           0.40      2233
   macro avg       0.39      0.40      0.39      2233
weighted avg       0.39      0.40      0.39      2233
```



We changed the learning rate once more to 0.0005

Initial Model with lr=0.0005

```
140/140 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.62      0.64      0.63       382
           1       0.33      0.02      0.03       381
           2       0.24      0.03      0.06       381
           3       0.31      0.54      0.40       382
           4       0.37      0.51      0.43       326
           5       0.44      0.75      0.55       381

    accuracy                           0.41      2233
   macro avg       0.39      0.42      0.35      2233
weighted avg       0.39      0.41      0.35      2233
```

The most confusing classes here are Fear 0.031,disgust 0.018 This model yields acc 0.40 and F1-score with weighted avg 0.35 and with macro average 0.35.

No significant approvement were achieved so we modified the architecture of the model.

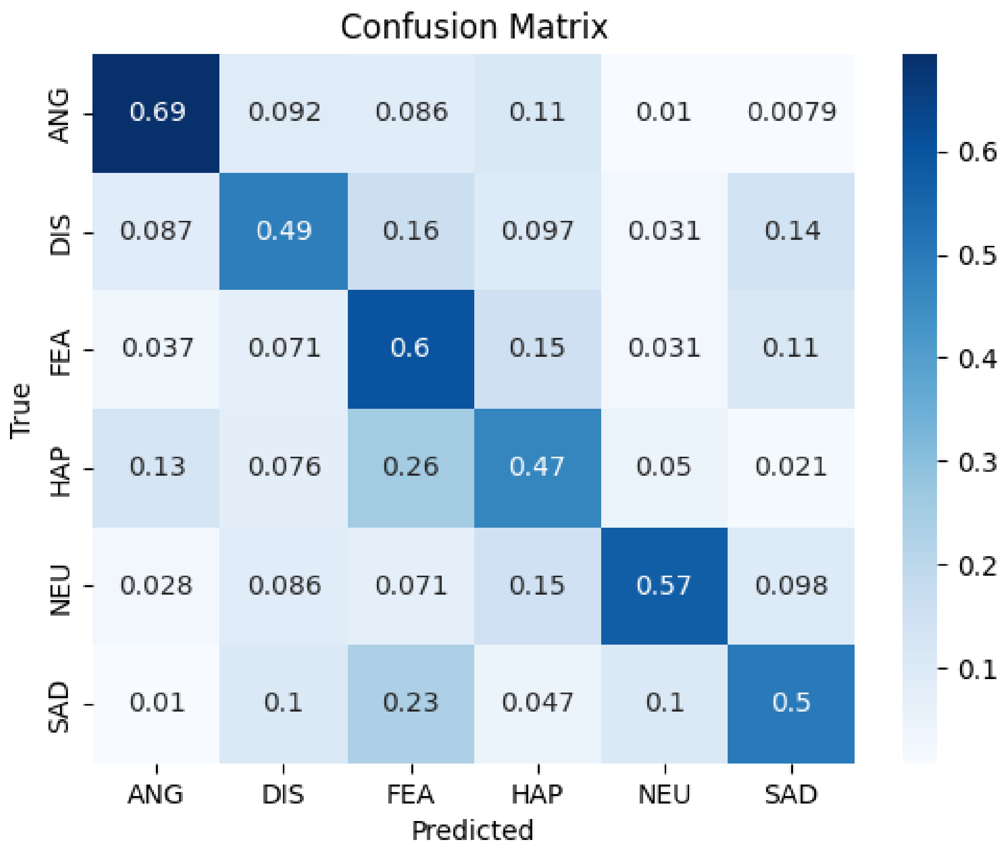
Second Model

```
def get_2d_conv_model(n):
    nclass = 6
    inp = Input(shape=(n,216,1))
    x = Convolution2D(32, (4,10), padding="same")(inp)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Convolution2D(32, (4,10), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Convolution2D(32, (4,10), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Convolution2D(32, (4,10), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Flatten()(x)
    x = Dense(64)(x)
    x = Dropout(rate=0.2)(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Dropout(rate=0.2)(x)

    out = Dense(nclass, activation=softmax)(x)
    model = models.Model(inputs=inp, outputs=out)

    opt = optimizers.Adam(0.001)
    model.compile(optimizer=opt, loss=losses.categorical_crossentropy, metrics=['acc',get_f1])
```

We changed the model architecture, including but not limited to changing the kernel size to 4x10 and added batch normalization following each conv2d layer. We also increased our use of conv2d.

Thes changes allowed us to improve our model substantially.

```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.71      0.69      0.70       382
           1       0.54      0.49      0.51       381
           2       0.43      0.60      0.50       381
           3       0.47      0.47      0.47       382
           4       0.68      0.57      0.62       326
           5       0.58      0.50      0.54       381

    accuracy                           0.55      2233
   macro avg       0.57      0.55      0.56      2233
weighted avg       0.57      0.55      0.56      2233
```



This is the best model with validation acc 0.58238 and testing acc 0.55 ,F1 score weighted and macro =0.56 The most confusing classes are HAP 0.47 and DIS 0.49

We then attempted to use a VGG16 like Model Architecture though it yielded poor results equal to that of random guessing.

We then tried another model architecture which yielded slightly better results than VGG16 but it ranked second to last.

Model 3:

```
def get_2d_conv_model():
    model = Sequential()
    model.add(Conv2D(32,8, 8, input_shape = (30,216,1),padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(32,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(32,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(64,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(64,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(64,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
    model.add(Dropout(0.5))

    model.add(Conv2D(64,8, 8,padding='same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

    model.add(Flatten())
    model.add(Dense(32,activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(6, activation = 'softmax'))


    opt=tensorflow.keras.optimizers.Adam(
    learning_rate=0.001
)
    model.compile(loss = 'categorical_crossentropy',optimizer =opt,metrics = ['accuracy'])
    return model
```

```
140/140 [==============================] - 1s 4ms/step
              precision    recall  f1-score   support

           0       0.60      0.58      0.59       382
           1       0.00      0.00      0.00       381
           2       0.00      0.00      0.00       381
           3       0.32      0.48      0.39       382
           4       0.33      0.79      0.46       326
           5       0.48      0.63      0.54       381

    accuracy                           0.40      2233
   macro avg       0.29      0.41      0.33      2233
weighted avg       0.29      0.40      0.33      2233
```

Fear and Disgust were the most confusing classes, with the model never choosing them for any predictions.

This leaves our second model architecture as the best model among the 2D models.