Presentaion Scoring Emotion

Engr. *Yamen Alkayyal*

27/9/2022

Contents

1 Introduction:

In this project we build Five deep learning classification models to classify the facial emption in to five categories :

1- Bordem.
2- Engagement.
3- Confusion.
4- Frustration.
5- Delight.

First we are training our models on the Daisee Dataset that is constructed of 4 classes of the above 5 mentioned ones and they are listed as follows:

1- Bordem.
2- Engagement.
3- Confusion.
4- Frustration.

Then through fine tuning our models to our custome dataset we are trying to intorduce the missing Delight class that is only present in our custom Dataset.
The Daisee Dataset is made up of 9068 video snippets captured from 112 users.
The Daisee Dataset used in this project can be found Here.
The Daisee Dataset is spliited into:

1- Train: 70%.
2- Validation: 15%.
3- Test: 15%.

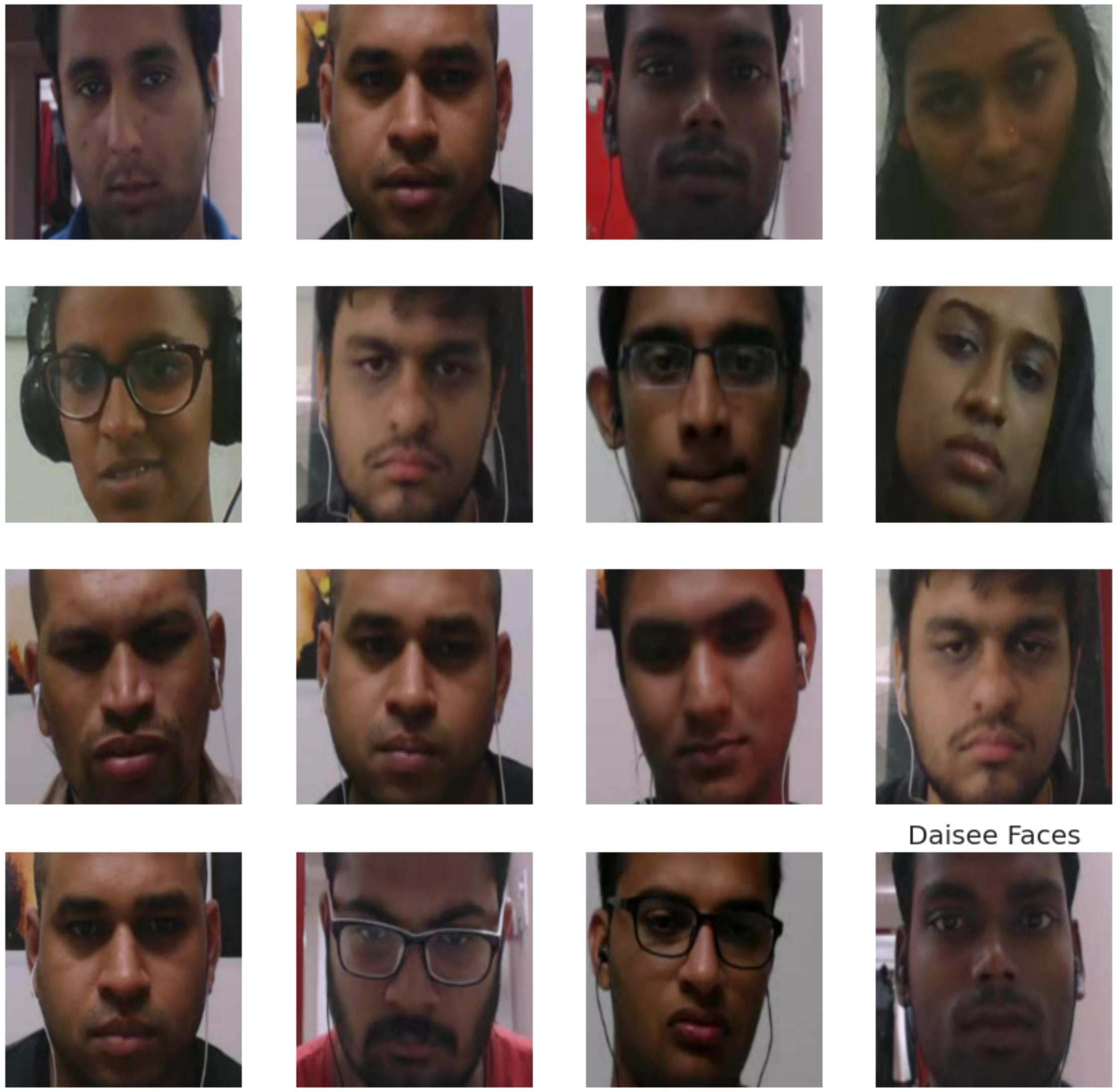

Figure 1: Samples from the Daisee dataset

2 Daisee Image Extraction:

Algorithm 1 Extract Faces

Input:

(1)*videos*: The original videos.

Output:

framesArray.
facesArray.
cords: face cordinates in image

```
start algorithm
for each video in videos do
    framesArray = extractFrames(video)
    for each frame in framesArray do
        cords = extractFaceCords(frame)
        face = cutImage([cords])
        facesArray = append(facesArray,face)
    end for
end for
end algorithm
```

Algorithm 2 link faces with labels

Input:

(1)$faces$: images cropped to show only the face of the participant.
(2)$labels$: dataframe of daisee dataset.

Output:

$numpyFaces\&labels$.
$face\&label$: each face image with it's corrosponding label.

```
start algorithm
for each faces,label in zip(faces,label) do
    face = toNumpy(face)
    label = toNumpy(label)
    face&label = toNumpy(link(face,label))
    numpyFaces&labels = append(numpyFaces&labels,face&label)
end for

end algorithm
```

3 Daisee Multi Class Evaluation:

Algorithm 3 Multi Task Lerning

Input:

(1)$numpyFaces\&labels$: train and test dataset.
(2)$pretrainedModel$.
(3)$pretrainedWieghts$.

Output:

$Model$.

start algorithm
Model = loadModel(pretrainedModel,pretrainedWeights)
Model = Model.append(fullyConnected)
Model = Model.append(Bordem: y1 = "sparseCategoricalCrossentropy")
Model = Model.append(Engagement: y2 = "sparseCategoricalCrossentropy")
Model = Model.append(Confusion: y3 = "sparseCategoricalCrossentropy")
Model = Model.append(Frustration: y4 = "sparseCategoricalCrossentropy")
train = model.fit(numpyFaces&labels)

end algorithm

	Xception	Inception	ResNet	MobileNet	EfficientNet
Bordem	0.40	0.40	0.44	0.43	0.46
Engagement	0.52	0.48	0.49	0.50	0.45
Confusion	0.67	0.67	0.67	0.67	0.67
Frustration	0.42	0.40	0.47	0.41	0.46

Table 1: Daisee Dataset Evaluation

4 Custom Dataset:

The Dataset is made up of 89 video snippets captured from 22 users.
The custom Dataset used in this project can be found Here.
The Custom Dataset is spliited into:

1- Train: 88%.
2- Validation: 12%.

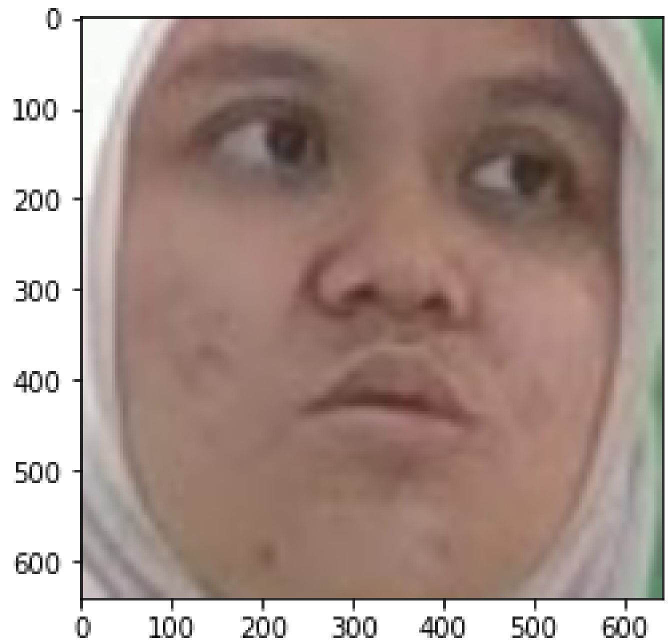

Figure 2: Sample from the Custom dataset

Algorithm 4 Add Delight Class

Input:

(1)*Model*: the model that has been fitted on the daisee dataset.

Output:

customModel.

fullyConnected: the last layer before multi task learning nodes

```
start algorithm
customModel = deepCopy(Model)
fullyConnected = customModel[-5:]
fullyConnected = append(fullyConnected,(Delight: y5 = 'sparseCategoricalCrossen-
tropy"))
customModel[-5:] = fullyConnected)

end algorithm
```

Algorithm 5 Custom Data Generator

Input:

(1)*videos*: custom videos.
(2)*videosLabels*:custom label files for each videos.
(3)*dataGenerator*:custom batches of data yielder that inputs one image from faces array and it's corrosponding csv provided by the assenbled csv of all labels.

Output:

faces:faces extracted from a single video.
facesDataset: all faces from all videos *labelsCsv*: label concatenated into one file.

```
start algorithm
for each video in videos do
    faces = extractFaces(video)
    facesDataset = append(facesDataset,faces)
end for
labelsCsv = csv.stack(videosLabels)
for n in batchSize do
    trainGenerator = dataGenerator.Generate(facesDataset[n],labelsCsv[n])
end for
train = model.fit(trainGenerator)

end algorithm
```

5 Five Classes Multi Class Evaluation:

	Xception	Inception	ResNet	MobileNet	EfficientNet
Bordem	0.54	0.57	0.22	0.36	0.67
Engagement	0.49	0.58	0.18	0.38	0.66
Confusion	0.47	0.59	0.28	0.35	0.67
Frustration	0.50	0.57	0.11	0.39	0.68
Delight	0.51	0.58	0.38	0.36	0.67

Table 2: Custom Data Evaluation

6 Classes F1 Scores:

Algorithm 6 one output only

Input:

(1)*Model*: the model that has been fitted on the custom classes dataset.
(2)*valGenerator*: custom generated validation data.

Output:

customData.
preds.

start algorithm
customModel = deepCopy(Model[:-6])
classificationLayer = Dense(5.softmax)
customModel = append(customModel,classificationLayer)
train = mode.fit(customData)
for valSample in valGenerator **do**
  pred = customModel.predict(valSample)
  preds = append(preds,pred)
end for

end algorithm

	Xception	Inception	ResNet	MobileNet	EfficientNet
Bordem	0.62	0.58	0.01	0.36	0.62
Engagement	0.84	0.78	0.54	0.55	0.81
Confusion	0.60	0.64	0.33	0.08	0.69
Frustration	0.52	0.51	0.34	0.27	0.62
Delight	0.38	0.08	0.00	0.00	0.44
accuracy	0.64	0.61	0.40	0.40	0.69
macro avg	0.59	0.52	0.25	0.25	0.64
weighted avg	0.60	0.59	0.30	0.33	0.68

Table 3: F1-Scores of each model implied on the 5 Classes

7 Classes Classification Reports:

- Inception Results:

	Precision	Recall	F1-Score	Support
Bordem	0.55	0.62	0.58	478
Engagement	0.71	0.86	0.78	748
Confusion	0.71	0.58	0.64	311
Frustration	0.49	0.52	0.51	406
Delight	0.23	0.05	0.08	235
accuracy			0.61	2178
macro avg	0.54	0.52	0.52	2178
weighted avg	0.58	0.61	0.59	2178

Table 4: F1-Scores of each model implied on the 5 Classes

- Xception Results:

	Precision	Recall	F1-Score	Support
Bordem	0.62	0.62	0.62	478
Engagement	0.85	0.83	0.84	748
Confusion	0.74	0.50	0.60	311
Frustration	0.52	0.52	0.52	406
Delight	0.32	0.47	0.38	235
accuracy			0.64	2178
macro avg	0.61	0.59	0.59	2178
weighted avg	0.66	0.64	0.65	2178

Table 5: F1-Scores of each model implied on the 5 Classes

- ResNet Results:

	Precision	Recall	F1-Score	Support
Bordem	0.21	0.01	0.01	478
Engagement	0.39	0.89	0.54	748
Confusion	0.48	0.25	0.33	311
Frustration	0.40	0.29	0.34	406
Delight	0.00	0.00	0.00	235
accuracy			0.40	2178
macro avg	0.30	0.29	0.25	2178
weighted avg	0.32	0.40	0.30	2178

Table 6: F1-Scores of each model implied on the 5 Classes

- MobileNet Results:

	Precision	Recall	F1-Score	Support
Bordem	0.64	0.25	0.36	478
Engagement	0.42	0.81	0.55	748
Confusion	0.76	0.04	0.08	311
Frustration	0.24	0.31	0.27	406
Delight	0.00	0.00	0.00	235
accuracy			0.40	2178
macro avg	0.41	0.28	0.25	2178
weighted avg	0.44	0.40	0.33	2178

Table 7: F1-Scores of each model implied on the 5 Classes

- EfficientNet Results:

	Precision	Recall	F1-Score	Support
Bordem	0.69	0.57	0.62	478
Engagement	0.72	0.93	0.81	748
Confusion	0.76	0.63	0.69	311
Frustration	0.61	0.63	0.62	406
Delight	0.58	0.35	0.44	235
accuracy			0.69	2178
macro avg	0.67	0.62	0.64	2178
weighted avg	0.68	0.69	0.68	2178

Table 8: F1-Scores of each model implied on the 5 Classes

8 Visual Results:

- Inception Results:

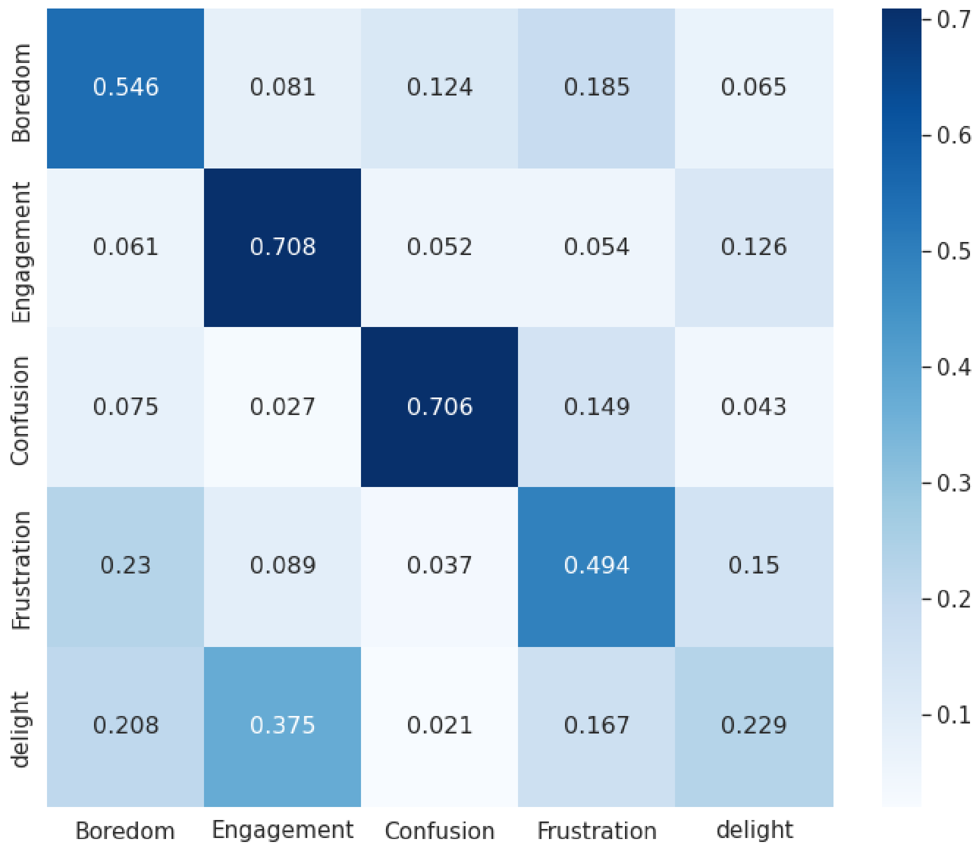

Figure 3: Xception Confusion Matrix

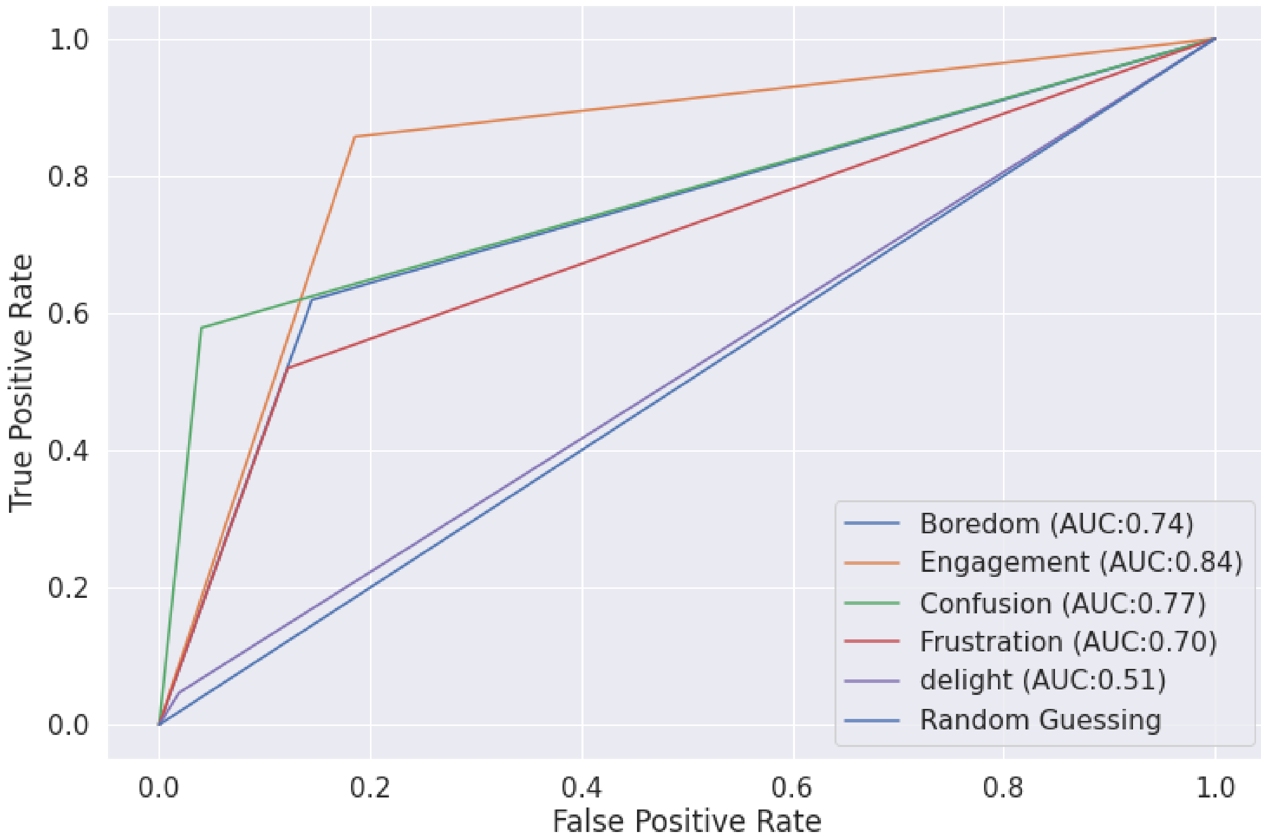

Figure 4: Xception ROC

- Xception Results:

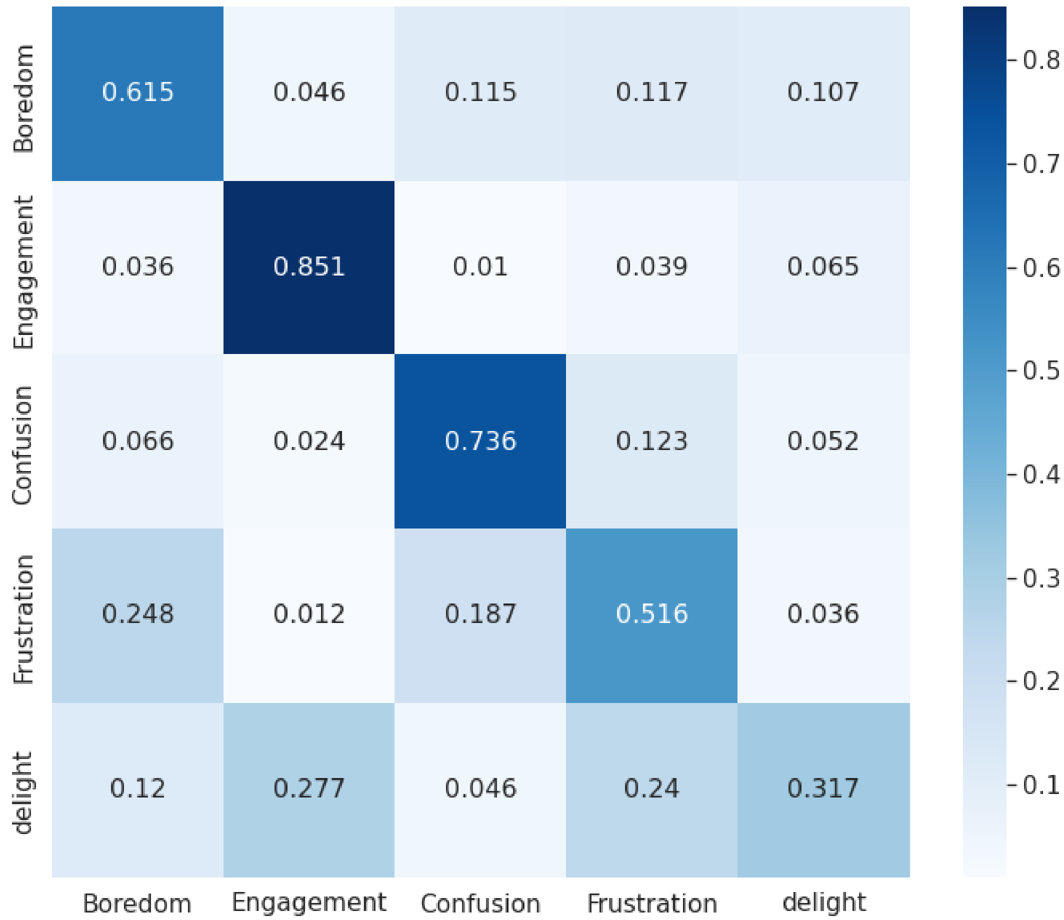

Figure 5: Xception Confusion Matrix

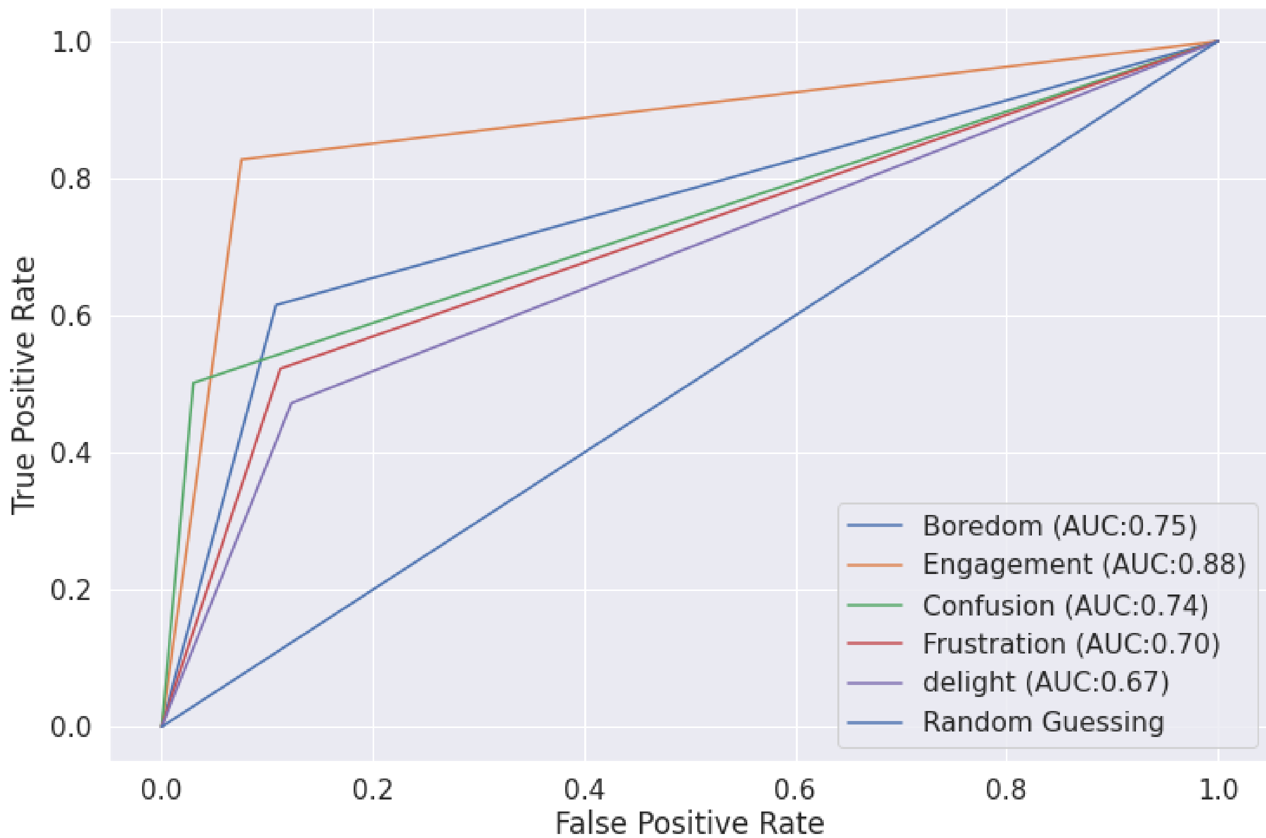

Figure 6: Xception ROC

- ResNet Results:

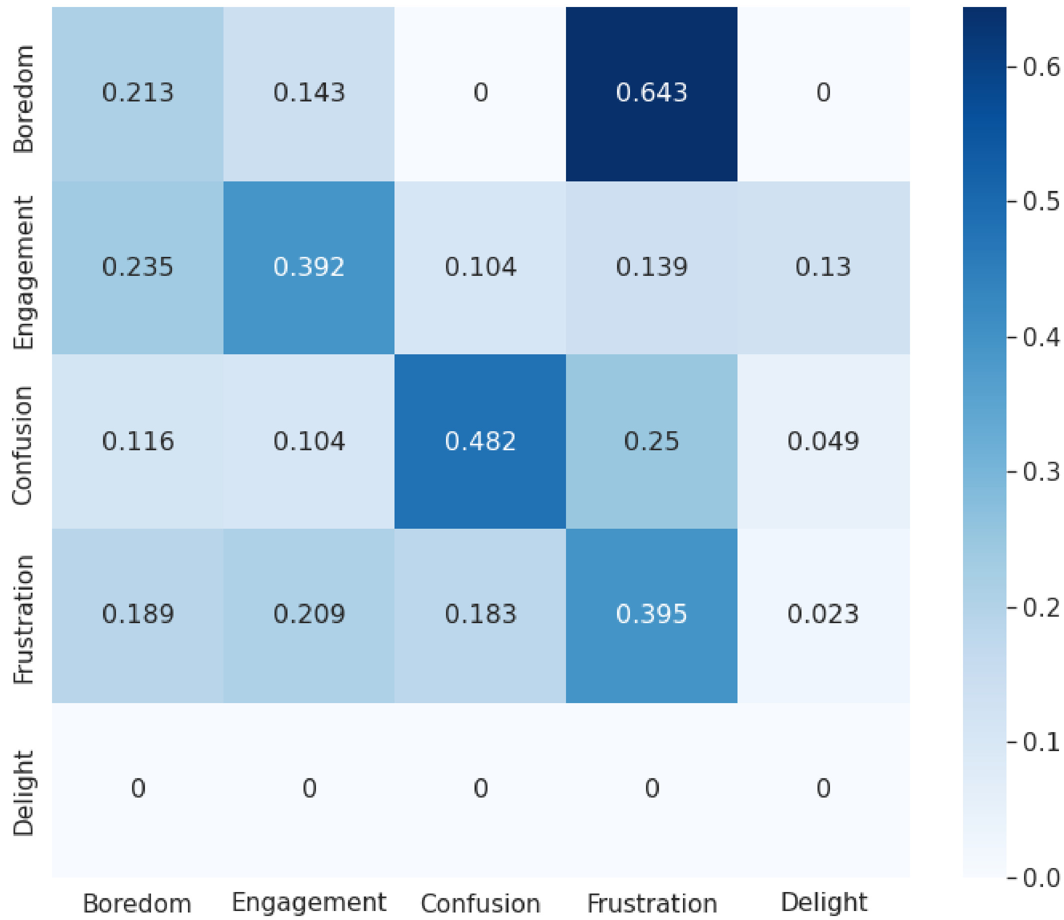

Figure 7: ResNet Confusion Matrix

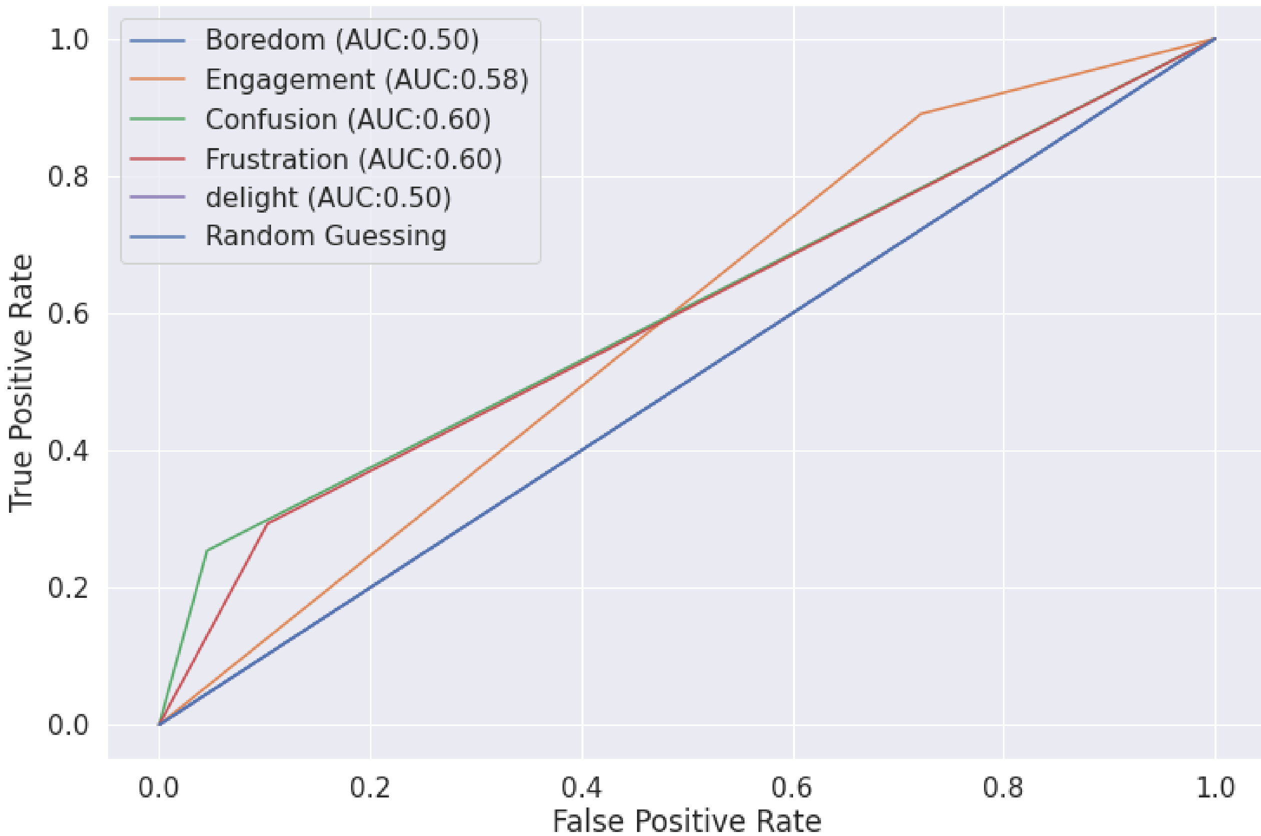

Figure 8: ResNet ROC

- MobileNet Results:

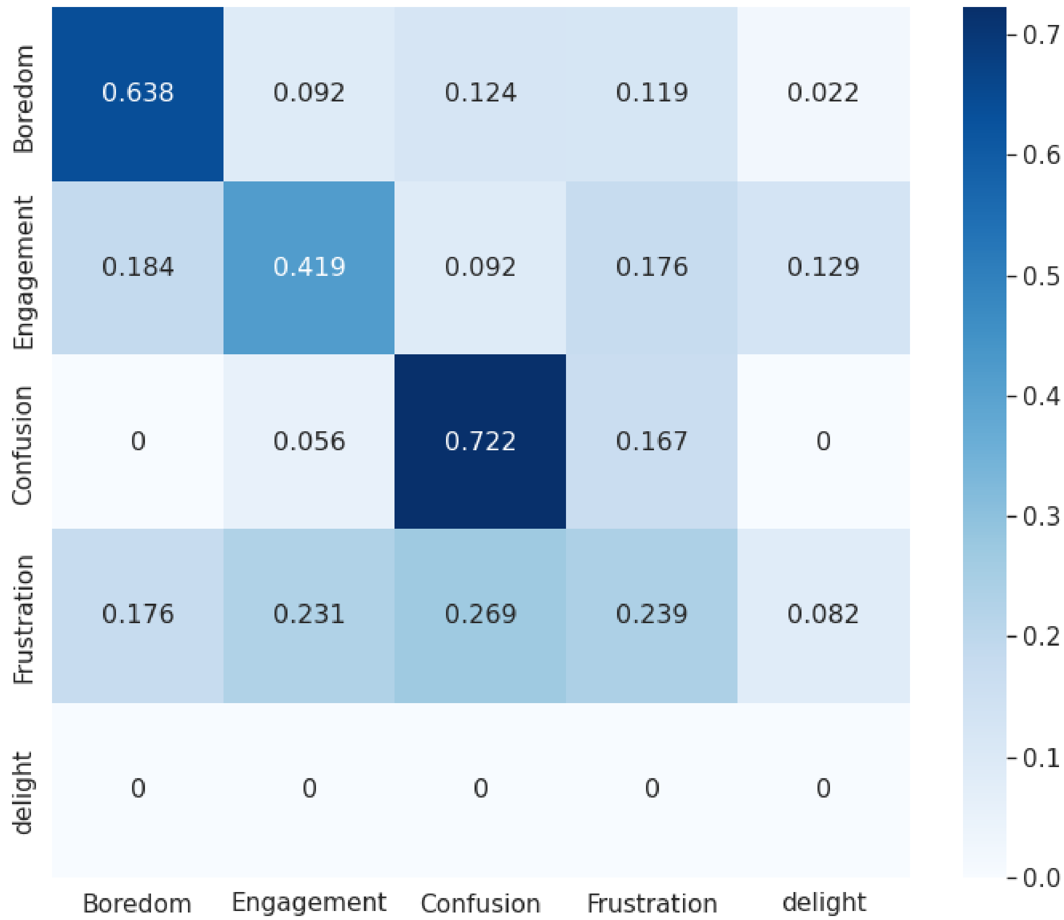

Figure 9: MobileNet Confusion Matrix

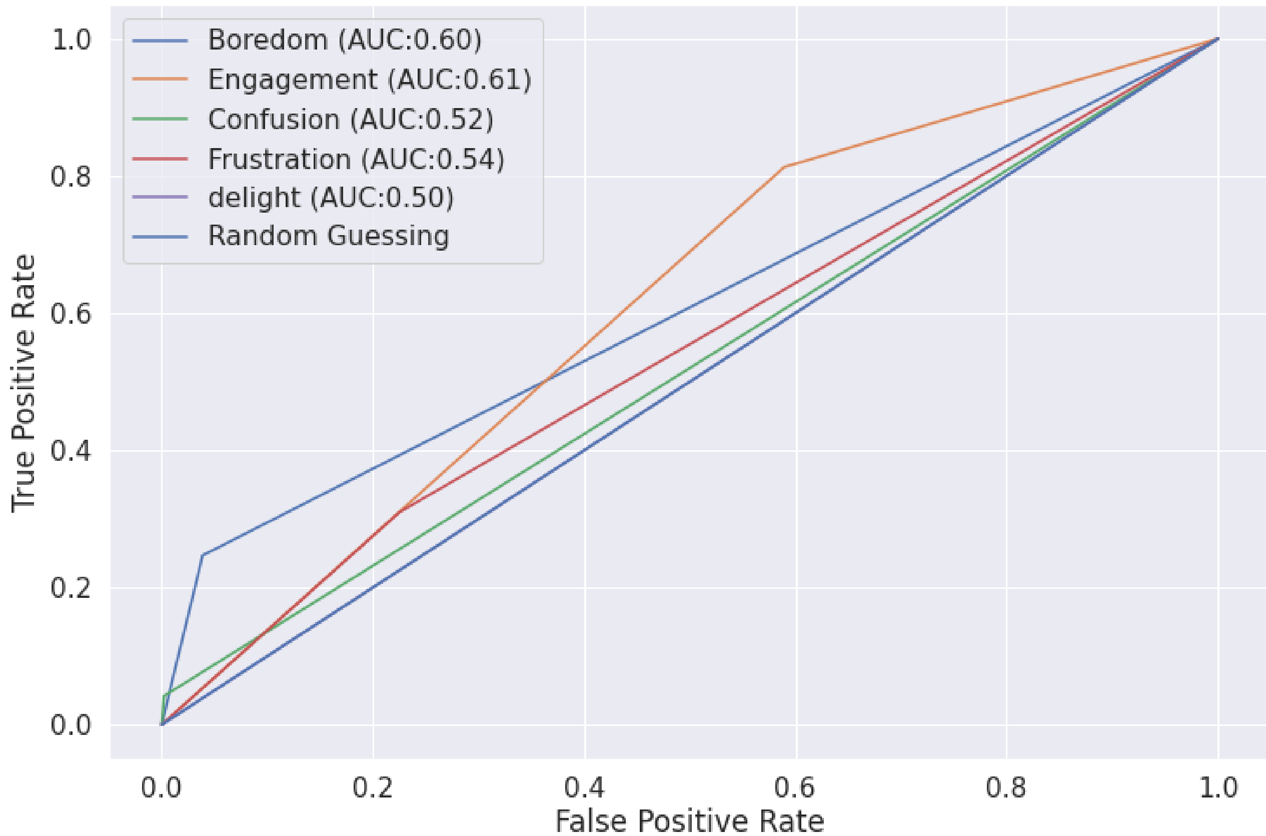

Figure 10: MobileNet ROC

- EfficientNet Results:

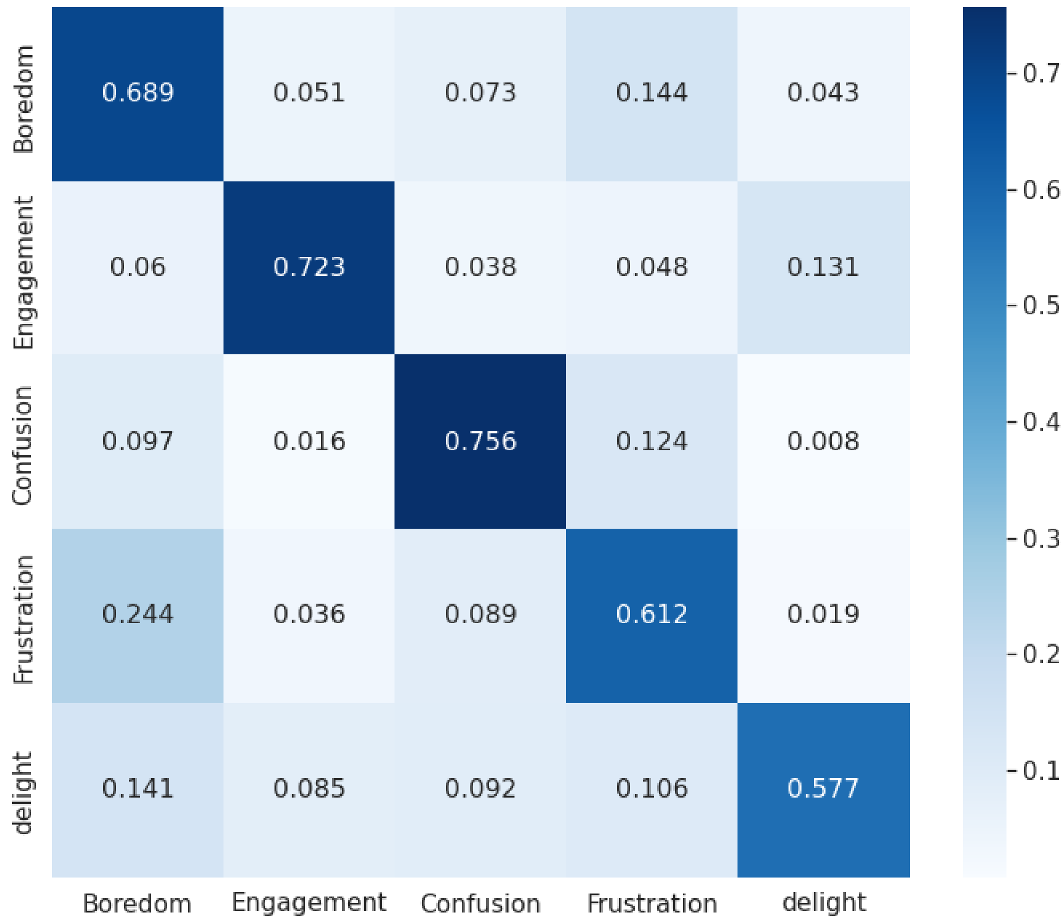

Figure 11: EfficientNet Confusion Matrix

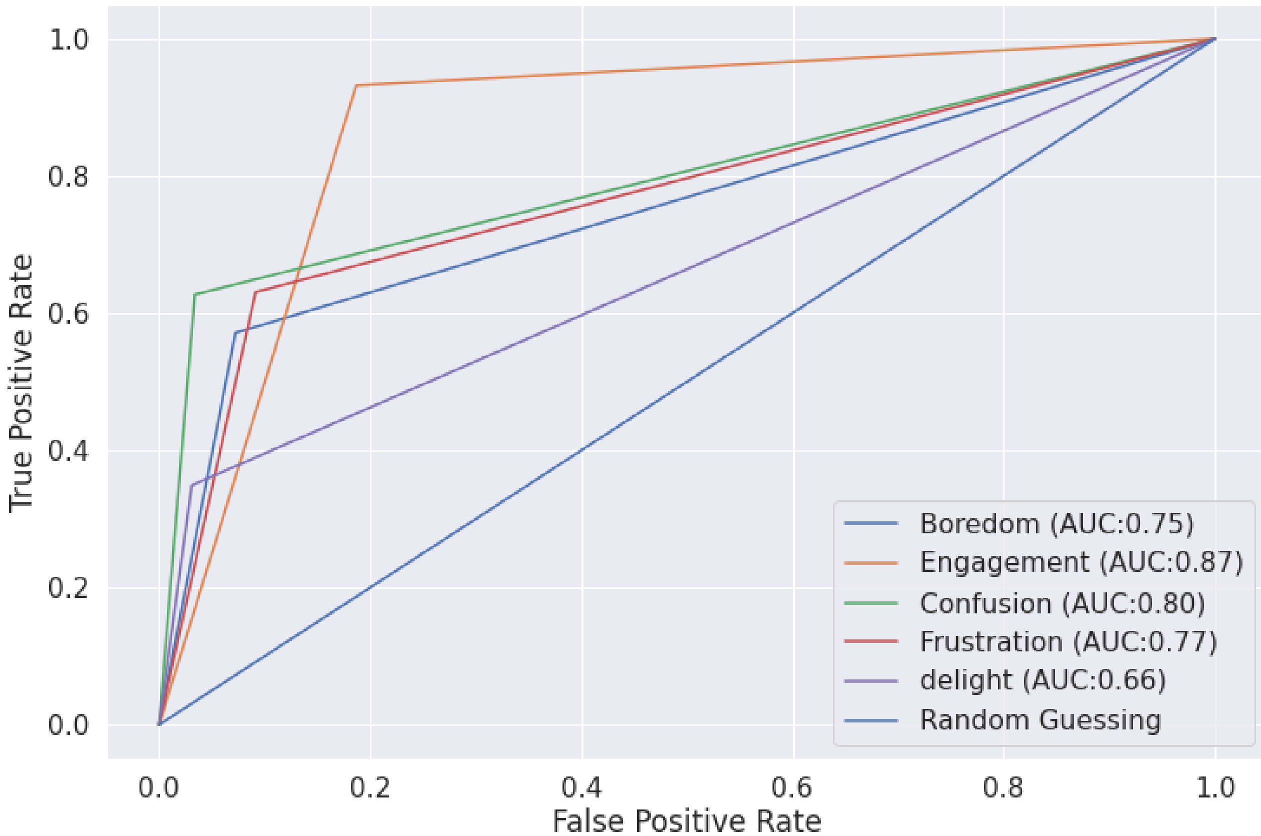

Figure 12: EfficientNet ROC