Gina Cody School of Engineering and Computer Science
Concordia University

COMP 6721 – Artificial Intelligence
Project Assignment Part-1

SNO	Student Name	Student ID	Specialisation
1	Roshini Chukkapalli	40198936	Data Specialist
2	Srujitha Yeruva	40232527	Evaluation Specialist
3	Sree Sneha Kothapalli	40235510	Training Specialist

Link to GitHub Repository: https://github.com/rochuk/COMP-6721---AI-ducation-analysis-project/tree/main

Dataset:

The objective of this project is to develop a Deep Learning Convolutional network using PyTorch. With this objective, we can analyse the images of students or people and categorize them into various classes. To achieve this objective, the classes we considered in this project are Neutral, Engaged/Focused, Bored/ Tired and Angry / Irritated.

Overview of the datasets:

The total number of images present in the dataset are 2680.

Neutral:

This facial expression is recognised when his face is usually relaxed and has no other emotional expression involved. There are a total of 750 images for this class including the training and test data. The training data has 625 images, and the test data has 125 images in total. Most of these images are frontal face shots and the images are of size 224*224 pixel. These images are in PNG format and this class contain images that are from various backgrounds. All these images are not uniform images and has still poses. The images in this class are squared images and focusses mainly on the facial expression of the person.

Engaged/Focused:

This facial expression is recognized when a student's face shows evidencing signs of active concentration, with sharp and attentive eyes. There are total of 550 images for this class including training and test data. Most of these images are frontal face shots and the images are of size 224*224 pixel. These images are in PNG format and this class contain images that are from various backgrounds. All these images are not uniform images and have still poses. The images in this class are squared images and focus mainly on the facial expression of the person.

Bored/Tired:

This facial expression is recognised when a person exhibiting signs of fatigue or disinterest, which can manifest through tired-looking eyes or absent-minded gazes. The dataset comprises 630 PNG images of excellent quality including training and test data. Most of them having a resolution of 1024x1024 pixels. It exhibits significant diversity in terms of age, ethnicity, and background settings, ensuring a broad representation of human characteristics. Additionally, the dataset provides comprehensive coverage of various accessories like eyeglasses, sunglasses, hats, and

more. These images were obtained by web crawling from Flickr, which means they may reflect the biases present on that platform.

Angry/Irritated:

This facial expression is recognised when his face has signs of agitation or displeasure which may result tightened facial muscles, narrow eyes, and a tight-lipped frown. There are a total of 750 images for this class including the training and test data. The training data has 625 images, and the test data has 125 images in total. Most of these images are frontal face shots and the are of size 224*224 pixel. These images are in PNG format and this class contain images of people from various backgrounds. All these images are not uniform images and has still poses. The images in this class are squared images and focusses mainly on the facial expression of the person.

Justification:

Neutral and Angry:

These classes are particularly taken from Kaggle. The reason for considering this dataset in Kaggle is because, these datasets include a wide variety of frontal shots and images with diverse backgrounds. On top of that, these images are classified images which reduces the work to manually sort them. These images do not contain duplicate images and are of good quality. This dataset also has a greater number of images in each class which allows us to choose the images wewant as our requirement should be a minimum of 2000 images in total. Although some images have more pixels and cannot be visualised properly, it has been taken care of by using light augmentation techniques.

Engaged: This dataset is a combination of images obtained from multiple sources on the internet. We have manually picked the relevant images from the kaggle dataset and istock (an online image library) mentioned in the source in below section, as we couldn't find a direct dataset. As we have picked the images manually from the mentioned sources, some images had to be converted into gray scale images and resized to uniform size. The images have been processed, including alignment and cropping, using the PIL, dlib libraries.

Bored: The choice of datasets was made to ensure that we have a diverse and balanced dataset forour project, which aims to classify facial expressions into different emotional states. The "Bored"

class is particularly relevant as it represents one of the mandatory emotions to be detected. The "Bored" images were selected from the Flickr-Faces-HQ (FFHQ) dataset, a comprehensive dataset of high-quality human faces. We chose this dataset because it provides a wide variety of facial expressions, including expressions of boredom. The challenge we encountered was in accurately labeling images with the correct class due to differences in class definitions among the source datasets. Furthermore, the images have been automatically processed, including alignment and cropping, using the dlib library.

Detailed information about images:

SNO	Class	Source Link of dataset	AuthorName	Liscence
1.	Neutral	https://www.kaggle.com/datasets/sudarshanvaidya/corrective-reannotation-of-fer-ck-kdef/data	Sudarshan Vaidya (Owner)	Unknown
2.	Angry	https://www.kaggle.com/datasets/sudarshanvaidya/corrective-reannotation-of-fer-ck-kdef/data	Sudarshan Vaidya (Owner)	Unknown
3.	Bored	https://github.com/NVlabs/ffhq-dataset	Tero Karras (NVIDIA), Samuli Laine (NVIDIA), Timo Aila (NVIDIA)	Flickr
4.	Engaged	https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer https://www.kaggle.com/code/hanimohamed/student-engagement-model/input istock https://github.com/NVlabs/ffhq-dataset	ARES Hani Mohamed Tero Karras (NVIDIA), Samuli Laine (NVIDIA), Timo Aila (NVIDIA)	CC0: Public Domain

Data Cleaning:

Data Cleaning is a process on which we remove or fix duplicates, incorrectly formatted, orientation of the dataset. This step is very crucial step to build a convoluted neural network. Here, in this project, we are collecting datasets from multiples sources so there are high chances that the data might be labelled incorrectly or may contain duplicate images. If this happens, then there are chances that the algorithm may provide incorrect or biased results. So, data cleaning is a very necessary step.

The images can have varied sizes, resolutions, and lighting issues. To make sure that the images were of same size and more robustness, in this project, after collecting the data from the dataset, apython script is written to clean the data.

Resizing the images:

All the images in the dataset have a constant size i.e., 256*256 pixels. This is performed by defining a transformation, a function transforms.resize() in which the size of the images are are passed as parameters. This is to ensure that the model is trained properly and gives the exact output.Greyscale conversion:

All the images in the dataset are converted to grey scale so that the facial expression of the datasetis more precise and can be clearly predicted. This is achieved by using the transforms library. Thefunction called is transforms.ToPILImage().

Light augmentation:

All the images in the dataset are light augmented. They are adjusted to a brightness factor of 0.2 and the contrast factor is 0.2. This is achieved by the transforms library. The function called is tranforms.ColorJitter().

After the data cleaning process is done, all the images are saved into a separate directory.

In the below figures, ‘a’ represents the original image whereas ‘b’ represents the data cleaned image with size being increased to 256*256, converted to a grey scale and has improved light i.e. brightness and contrast of the image is improved.

a. Original Image

b. Data Cleaned Image.

CNN Architecture:

We developed three different models for training.

The first model is the main architecture, and the other two models are the variants of the mainarchitecture.

The entire dataset is preprocessed by resizing the images to 224*224 pixels, converted them togrey scale, performed a random horizontal flip and randomly cropped the images in the dataset. The dataset is then split into training data, validation data and test data using train_test_split tool.The split ratios of the dataset are 70:15:15 for training data, validation data and test datarespectively.

Main Architecture:

The input layer is a single channel input because the images in the dataset are greyscale images. The main architecture has 4 convolutional layers. The structure of convolutional layer is nn.Conv2d(1,32,3,padding=1,stride=1)

Convolutional layer -1:

This layer has 1 input channel, 32 output channels, 3*3 kernel size with padding and stride values as 1. After the first convolutional layer, batch normalisation is applied to normalise the activations with a normalisation value of 32.

Convolutional layer - 2:

This layer has 32 input channels, 32 output channels, 3*3 kernel size with padding and stride values as 1. After the second convolutional layer, batch normalisation is applied to normalise the activations with a normalisation value of 32.

Convolutional layer - 3:

This layer has 32 input channels, 64 output channels, 3*3 kernel size with padding and stride values as 1. After the third convolutional layer, batch normalisation is applied to normalise the activations with a normalisation value of 64.

Convolutional layer - 4:

This layer has 32 input channels, 64 output channels, 3*3 kernel size with padding and stride values as 1. After the third convolutional layer, batch normalisation is applied to normalise the activations with a normalisation value of 64.

Maxpool layer:

There's one max pooling layer of size 2*2. This is used to decrease the spatial dimensions of the input dataset while retaining the most important information regarding dataset.

Fully connected layer:

Linear layer:

The input size of this layer is based on the dimensions after the last convolutional layer i.e., the 4^{th} layer. So, size of this layer will be 64 * (224//4) * (224//4).

The activation function used is Leaky Relu which introduces a small, non-zero slope values to negative inputs when the input values are less than zero.

Variant – 1:

The variant -1 is slightly like the main architecture but the kernel size is increased to 5*5 and padding is increased to 2.

Increased Kernel Size (5*5):

The reason behind increasing the kernel size is to capture the information from a larger area of theinput image which leads to learning more complex and higher-level features. If the kernel size is increased, the number of parameters in the network also increases. This allows the model to learnmore intricate details of the data. It can also lead to smoothing effect thereby reducing the sensitivity to small variations in the input dataset. Although, it has benefits, it comes with increased cost and larger training times.

Increased padding size (to 2):

Increase in kernel size can lead to the convolution network reducing the spatial dimensions of thefeature maps. Padding is increased to control this reduction. If the padding size is increased, it is guaranteed that the convolution layer does not trim the edges of the input, retaining the spatial details of dataset.

Increasing the kernel size can lead to overfitting

Variant – 2:

The variant -2 is also slightly like main architecture but has an additional convolutional layer.

It has all 4 convolutional layers in addition to another layer.

Convolutional Layer -5:

This layer has 64 input channels, 128 output channels, 3*3 kernel size with padding and stride values as 1. After the third convolutional layer, batch normalisation is applied to normalise the activations with a normalisation value of 128.

Fully connected layer:

Linear layer:

The input size of this layer is based on the dimensions after the last convolutional layer i.e., the 5^{th}layer. So, size of this layer will be 128 * (224//8) * (224//8).

Reason:

The reason behind adding an additional layer to the network is because it increases the model's capacity to learn more hierarchical features from the dataset as each layer helps in capturing different levels of abstraction and this layer allows the convolutional network to learn more complex and discriminative features.

Regularization techniques:

To avoid over fitting of the data, all the three variants are developed with regularisation techniques.We used two techniques to develop the model architecture.

1. Early stopping:

Early stopping is a technique in which the validation loss is monitored to stop training process. If the validation loss stops decreasing or starts to increase, the training process is stopped. It also helps to ensure that the model is not trained for many epochs which lead to overfitting. In the mainarchitecture and variant – 2, if the validation loss stops decreasing or starts to increase for 5 consecutive epochs, then the training process is halted. Similarly, for variant -1, for 3 consecutiveepochs, if the validation loss stops decreasing or starts to increase, the training process is halted.

2. Batch Normalisation:

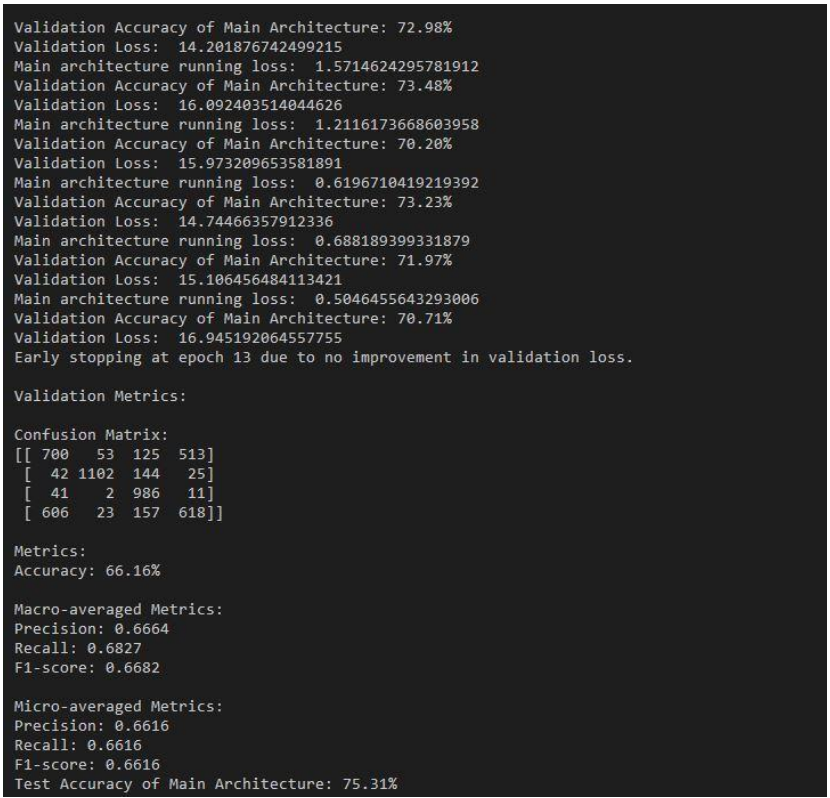
Although it is primarily used for stabilising and accelerating the training dataset, it can be used as regularization technique as it introduces the noise during training.

Deviations found in variants:

Main Architecture:

The accuracy of main architecture is recorded as 75.31%.

Although the number of epochs are 25, the training process is halted at 13th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 75.31%.



Variant -1 (variation in kernel size):

The accuracy of variant -1 architecture is recorded as 63.51%.

Although the number of epochs is 25, the training process is halted at 13th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 65.51%.

```
Validation Loss:  12.261216981070381
Variant-1 running loss :  0.7436931824375843
Validation Accuracy of Variant-1 Architecture: 65.91%
Validation Loss:  10.118653910500663
Variant-1 running loss :  0.47991355727747853
Validation Accuracy of Variant-1 Architecture: 64.90%
Validation Loss:  11.7464051927839
Variant-1 running loss :  0.8117949225265404
Validation Accuracy of Variant-1 Architecture: 61.62%
Validation Loss:  13.064483097621373
Variant-1 running loss :  0.41868221384055654
Validation Accuracy of Variant-1 Architecture: 64.14%
Validation Loss:  12.247114998953682
Early stopping at epoch 13 due to no improvement in validation loss.

Validation Metrics:

Confusion Matrix:
[[622  23 177 569]
 [ 64 888 287  74]
 [ 74 151 651 164]
 [426  13 185 780]]

Metrics:
Accuracy: 57.13%

Macro-averaged Metrics:
Precision: 0.5857
Recall: 0.5762
F1-score: 0.5761

Micro-averaged Metrics:
Precision: 0.5713
Recall: 0.5713
F1-score: 0.5713
Test Accuracy of Variant-1 Architecture: 63.51%
```

Variant -2 (Added an addition convolutional layer):

The accuracy of variant -2 architecture is recorded as 74.81%.

Although the number of epochs is 25, the training process is halted at 20th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 74.81%.

```
variant-2 running loss :  0.5207907576756231
Validation Accuracy of variant-2 Architecture: 70.71%
Validation Loss:  6.033590112413679
variant-2 running loss :  0.4302416657065523
Validation Accuracy of variant-2 Architecture: 75.76%
Validation Loss:  4.1988942963736395
variant-2 running loss :  0.18774366950423554
Validation Accuracy of variant-2 Architecture: 74.24%
Validation Loss:  4.299737385341099
variant-2 running loss :  0.32420131133805064
Validation Accuracy of variant-2 Architecture: 74.75%
Validation Loss:  4.503694602421352
variant-2 running loss :  0.31552612707661143
Validation Accuracy of variant-2 Architecture: 74.49%
Validation Loss:  4.41340218271528
variant-2 running loss :  0.44259306404273957
Validation Accuracy of variant-2 Architecture: 74.75%
Validation Loss:  6.078529426029751
variant-2 running loss :  0.3116293721511189
Validation Accuracy of variant-2 Architecture: 73.99%
Validation Loss:  5.003864390509469
Early stopping at epoch 20 due to no improvement in validation loss.

Validation Metrics:

Confusion Matrix:
[[1124   58  123  835]
 [ 218 1586  125   91]
 [ 171    2 1358   69]
 [ 823   45  164 1128]]

Metrics:
Accuracy: 65.61%

Macro-averaged Metrics:
Precision: 0.6794
Recall: 0.6703
F1-score: 0.6724

Micro-averaged Metrics:
Precision: 0.6561
Recall: 0.6561
F1-score: 0.6561
Test Accuracy of Variant-2 Architecture: 74.81%
```

Training Process:

Epochs:

The number of epochs for all three variants are set to 25. However, early stopping technique isused in case there is no progress in validation loss.

Learning rate:

The learning rate determines the size of the step the optimizer takes while run along with an optimization algorithm. The learning rate is set to 0.01. The reason behind choosing this learning rate is, the value is neither too high which can result in convergence issues, nor too low which can result in slowing down the training process.

Loss function:

The loss function used here is CrossEntropyLoss as it is suitable for multi-class classification tasks.It is imported from pyTorch library. It measures the performance of the classification with output being a probability value between 0 and 1. It is mainly used to when the task is to classify one class out of multiple classes.

Patience:

This is a hyperparameter used in the training process. The functionality of patience parameter is monitoring the validation loss of a model. If there is no improvement, then the training process ishalted. The value is set to 5 for main architecture and variant -2 and 3 for variant -1.

Optimization algorithm:

The optimization algorithm incorporated in the project Adam Optimizer. This algorithm is well known for its adaptive learning rates. It adjusts the learning rates of each parameter individually based on the previous gradients. It is highly efficient and requires less memory. This technique also converges faster than the other optimization techniques which can help the model develop and experiment quicker.

Regularization techniques:

To avoid over fitting of the data, all the three variants are developed with regularisation techniques.We used two techniques to develop the model architecture.

1. Early stopping:

Early stopping is a technique in which the validation loss is monitored to stop training process. If the validation loss stops decreasing or starts to increase, the training process is stopped. It also helps to ensure that the model is not trained for many epochs which lead to overfitting. In the mainarchitecture and variant – 2, if the validation loss stops decreasing or starts to increase for 5 consecutive epochs, then the training process is halted. Similarly, for variant -1, for 3 consecutiveepochs, if the validation loss stops decreasing or starts to increase, the training process is halted.

2. Batch Normalisation:

Although it is primarily used for stabilising and accelerating the training dataset, it can be used as regularization technique as it introduces the noise during training.

Deviations found in variants:

Main Architecture:

The accuracy of main architecture is recorded as 75.31%.

Although the number of epochs are 25, the training process is halted at 13th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 75.31%.

```
Validation Accuracy of Main Architecture: 72.98%
Validation Loss:  14.201876742499215
Main architecture running loss:  1.5714624295781912
Validation Accuracy of Main Architecture: 73.48%
Validation Loss:  16.092403514044626
Main architecture running loss:  1.2116173668603958
Validation Accuracy of Main Architecture: 70.20%
Validation Loss:  15.973209653581891
Main architecture running loss:  0.6196710419219392
Validation Accuracy of Main Architecture: 73.23%
Validation Loss:  14.74466357912336
Main architecture running loss:  0.688189399331879
Validation Accuracy of Main Architecture: 71.97%
Validation Loss:  15.106456484113421
Main architecture running loss:  0.5046455643293006
Validation Accuracy of Main Architecture: 70.71%
Validation Loss:  16.945192064557755
Early stopping at epoch 13 due to no improvement in validation loss.

Validation Metrics:

Confusion Matrix:
[[ 700   53  125  513]
 [  42 1102  144   25]
 [  41    2  986   11]
 [ 606   23  157  618]]

Metrics:
Accuracy: 66.16%

Macro-averaged Metrics:
Precision: 0.6664
Recall: 0.6827
F1-score: 0.6682

Micro-averaged Metrics:
Precision: 0.6616
Recall: 0.6616
F1-score: 0.6616
Test Accuracy of Main Architecture: 75.31%
```

Variant -1 (variation in kernel size):

The accuracy of variant -1 architecture is recorded as 63.51%.

Although the number of epochs is 25, the training process is halted at 13th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 65.51%.

```
Validation Loss:  12.261216981070381
Variant-1 running loss :  0.7436931824375843
Validation Accuracy of Variant-1 Architecture: 65.91%
Validation Loss:  10.118653910500663
Variant-1 running loss :  0.47991355727747853
Validation Accuracy of Variant-1 Architecture: 64.90%
Validation Loss:  11.7464051927839
Variant-1 running loss :  0.8117949225265404
Validation Accuracy of Variant-1 Architecture: 61.62%
Validation Loss:  13.064483097621373
Variant-1 running loss :  0.41868221384055654
Validation Accuracy of Variant-1 Architecture: 64.14%
Validation Loss:  12.247114998953682
Early stopping at epoch 13 due to no improvement in validation loss.

Validation Metrics:

Confusion Matrix:
[[622  23 177 569]
 [ 64 888 287  74]
 [ 74 151 651 164]
 [426  13 185 780]]

Metrics:
Accuracy: 57.13%

Macro-averaged Metrics:
Precision: 0.5857
Recall: 0.5762
F1-score: 0.5761

Micro-averaged Metrics:
Precision: 0.5713
Recall: 0.5713
F1-score: 0.5713
Test Accuracy of Variant-1 Architecture: 63.51%
```

Variant -2 (Added an addition convolutional layer):

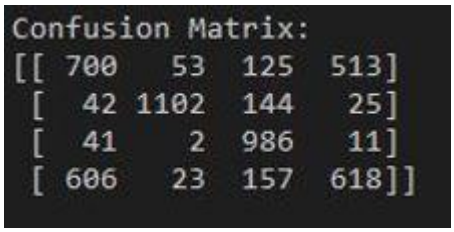
The accuracy of variant -2 architecture is recorded as 74.81%.

Although the number of epochs is 25, the training process is halted at 20th epoch as there is no improvement in the validation loss. The best model based on validation accuracy is saved and loaded to compute the accuracy of test data which gives the accuracy as 74.81%.

```
variant-2 running loss :  0.5207907576756231
Validation Accuracy of variant-2 Architecture: 70.71%
Validation Loss:  6.033590112413679
variant-2 running loss :  0.4302416657065523
Validation Accuracy of variant-2 Architecture: 75.76%
Validation Loss:  4.1988942963736395
variant-2 running loss :  0.18774366950423554
Validation Accuracy of variant-2 Architecture: 74.24%
Validation Loss:  4.299737385341099
variant-2 running loss :  0.32420131133805064
Validation Accuracy of variant-2 Architecture: 74.75%
Validation Loss:  4.503694602421352
variant-2 running loss :  0.31552612707661143
Validation Accuracy of variant-2 Architecture: 74.49%
Validation Loss:  4.41340218271528
variant-2 running loss :  0.44259306404273957
Validation Accuracy of variant-2 Architecture: 74.75%
Validation Loss:  6.078529426029751
variant-2 running loss :  0.3116293721511189
Validation Accuracy of variant-2 Architecture: 73.99%
Validation Loss:  5.003864390509469
Early stopping at epoch 20 due to no improvement in validation loss.

Validation Metrics:

Confusion Matrix:
[[1124   58  123  835]
 [ 218 1586  125   91]
 [ 171    2 1358   69]
 [ 823   45  164 1128]]

Metrics:
Accuracy: 65.61%

Macro-averaged Metrics:
Precision: 0.6794
Recall: 0.6703
F1-score: 0.6724

Micro-averaged Metrics:
Precision: 0.6561
Recall: 0.6561
F1-score: 0.6561
Test Accuracy of Variant-2 Architecture: 74.81%
```

Evaluation:

Confusion Matrix of Main Architecture:



- **Rows:** Each row corresponds to a predicted class.
- **Columns:** Each column corresponds to an actual class.

- **Row 1 (Angry):**
    - 700 instances of class Angry were correctly predicted as Angry (True Positive).
    - 53 instances of class Bored were mistakenly predicted as Angry (False Positive).
    - 125 instances of class Engaged were mistakenly predicted as Angry (False Positive).
    - 513 instances of class Neutral were mistakenly predicted as Angry (False Positive).
- **Row 2 (Bored):**
    - 1102 instances of class Bored were correctly predicted as Bored (True Positive).
    - 42 instances of class Angry were mistakenly predicted as Bored (False Positive).
    - 144 instances of class Engaged were mistakenly predicted as Bored (False Positive).
    - 25 instances of class Neutral were mistakenly predicted as Bored (False Positive).
- **Row 3 (Engaged):**
    - 986 instances of class Engaged were correctly predicted as Engaged (True Positive).
    - 41 instances of class Angry were mistakenly predicted as Engaged (False Positive).
    - 2 instances of class Bored were mistakenly predicted as Engaged (False Positive).
    - 11 instances of class Neutral were mistakenly predicted as Engaged (False Positive).

- **Row 4 (Neutral):**
    - 618 instances of class Neutral were correctly predicted as Neutral (True Positive).
    - 606 instances of class Angry were mistakenly predicted as Neutral (False Positive).
    - 23 instances of class Bored were mistakenly predicted as Neutral (False Positive).
    - 157 instances of class Engaged were mistakenly predicted as Neutral (False Positive).

Confusion Matrix of Variant-1:

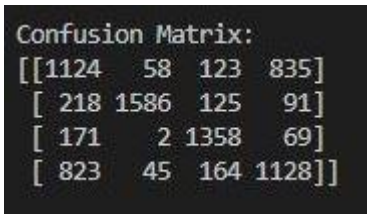
```
Confusion Matrix:
[[622  23 177 569]
 [ 64 888 287  74]
 [ 74 151 651 164]
 [426  13 185 780]]
```

- **Row 1 (Angry):**
    - 622 instances of class Angry were correctly predicted as Angry (True Positive).
    - 23 instances of class Bored were mistakenly predicted as Angry (False Positive).
    - 177 instances of class Engaged were mistakenly predicted as Angry (False Positive).
    - 569 instances of class Neutral were mistakenly predicted as Angry (False Positive).
- **Row 2 (Bored):**
    - 888 instances of class Bored were correctly predicted as Bored (True Positive).
    - 64 instances of class Angry were mistakenly predicted as Bored (False Positive).
    - 287 instances of class Engaged were mistakenly predicted as Bored (False Positive).
    - 74 instances of class Neutral were mistakenly predicted as Bored (False Positive).
- **Row 3 (Engaged):**
    - 651 instances of class Engaged were correctly predicted as Engaged (True Positive).
    - 74 instances of class Angry were mistakenly predicted as Engaged (False Positive).

- 151 instances of class Bored were mistakenly predicted as Engaged (False Positive).
- 164 instances of class Neutral were mistakenly predicted as Engaged (False Positive).

- **Row 4 (Neutral):**
  - 780 instances of class Neutral were correctly predicted as Neutral (True Positive).
  - 426 instances of class Angry were mistakenly predicted as Neutral (False Positive).
  - 13 instances of class Bored were mistakenly predicted as Neutral (False Positive).
  - 185 instances of class Engaged were mistakenly predicted as Neutral (False Positive).

Confusion Matrix of Variant-2:



- **Row 1 (Angry):**
  - 1124 instances of class Angry were correctly predicted as Angry (True Positive).
  - 58 instances of class Bored were mistakenly predicted as Angry (False Positive).
  - 123 instances of class Engaged were mistakenly predicted as Angry (False Positive).
  - 835 instances of class Neutral were mistakenly predicted as Angry (False Positive).
- **Row 2 (Bored):**
  - 1586 instances of class Bored were correctly predicted as Bored (True Positive).
  - 218 instances of class Angry were mistakenly predicted as Bored (False Positive).
  - 125 instances of class Engaged were mistakenly predicted as Bored (False Positive).
  - 91 instances of class Neutral were mistakenly predicted as Bored (False Positive).

- **Row 3 (Engaged):**
    - 1358 instances of class Engaged were correctly predicted as Engaged (True Positive).
    - 171 instances of class Angry were mistakenly predicted as Engaged (False Positive).
    - 2 instances of class Bored were mistakenly predicted as Engaged (False Positive).
    - 69 instances of class Neutral were mistakenly predicted as Engaged (False Positive).
- **Row 4 (Neutral):**
    - 1128 instances of class Neutral were correctly predicted as Neutral (True Positive).
    - 823 instances of class Angry were mistakenly predicted as Neutral (False Positive).
    - 45 instances of class Bored were mistakenly predicted as Neutral (False Positive).
    - 164 instances of class Engaged were mistakenly predicted as Neutral (False Positive).

The Performance Metrics Of The Main Model And Its Two Variants:

Model	Macro			Micro			Accuracy
	P	R	F	P	R	F	
Main Model	0.664	0.6827	0.6682	0.6616	0.6616	0.6616	75.31%
Variant 1	0.5857	0.5762	0.5761	0.5713	0.5713	0.5713	63.51%
Variant 2	0.6794	0.6703	0.6724	0.6561	0.6561	0.6561	74.81%

Insights:

1. **Accuracy:**
    - Main Model has the highest accuracy (75.31%), indicating overall better performance on the test set.
    - Variant 1 has the lowest accuracy (63.51%).
2. **Precision:**

- Main Model has the highest precision in both macro and micro averages.
- Variant 2 has the second-highest precision, while Variant 1 has the lowest precision.

3. **Recall:**
    - Main Model has the highest recall in both macro and micro averages.
    - Variant 2 has the second-highest recall, while Variant 1 has the lowest recall.
4. **F1-Score:**
    - Main Model has the highest F1-score in both macro and micro averages.
    - Variant 2 has the second-highest F1-score, while Variant 1 has the lowest F1-score.

Implications:

- A higher recall suggests that the Main Model is better at capturing positive instances, which is important in facial image analysis to minimize false negatives (missing actual positive instances).
- Higher precision in the Main Model indicates that when it predicts a class, it is more likely to be correct, reducing false positives.
- Variant 1, while having lower accuracy, may have a balance of precision and recall suitable for specific use cases or trade-offs between false positives and false negatives.
- Variant 2 shows a good balance, but its performance is slightly below the Main Model.

In facial image analysis, the choice between precision and recall depends on the specific application. For example:

- High precision is crucial in applications where false positives are costly (e.g., identifying security threats).
- High recall is important when missing positive instances (false negatives) is undesirable (e.g., medical diagnosis).

The Main Model seems to strike a good balance between precision and recall, making it a strong performer in this context

K-Fold

Fold-1

```
Main architecture running loss:  0.3630303602028442
Validation Accuracy of Main Architecture: 78.49%
Validation Loss:  1.0191724002361298
Early stopping at epoch 23 due to no improvement in validation loss.

Validation Metrics for Fold 1:

Metrics:
Accuracy: 73.54%

Macro-averaged Metrics:
Precision: 0.7351
Recall: 0.7461
F1-score: 0.7380

Micro-averaged Metrics:
Precision: 0.7354
Recall: 0.7354
F1-score: 0.7354
Confusion Matrix for Fold 1:
[[ 725   49   65  679]
 [  26 1536   65   52]
 [   5    3 1155   33]
 [ 507   31   98 1066]]
```

Fold-2

```
Validation Loss:  0.4678534399219643
Early stopping at epoch 9 due to no improvement in validation loss.

Validation Metrics for Fold 2:

Metrics:
Accuracy: 85.79%

Macro-averaged Metrics:
Precision: 0.8766
Recall: 0.8778
F1-score: 0.8768

Micro-averaged Metrics:
Precision: 0.8579
Recall: 0.8579
F1-score: 0.8579
Confusion Matrix for Fold 2:
[[538   0   5 186]
 [  3 492   0   0]
 [  0   0 504   0]
 [143   1   1 512]]
```

Fold-3

```
Validation Accuracy of Main Architecture: 87.92%
Validation Loss:  0.3085898760706186
Early stopping at epoch 6 due to no improvement in validation loss.

Validation Metrics for Fold 3:

Metrics:
Accuracy: 86.60%

Macro-averaged Metrics:
Precision: 0.8739
Recall: 0.8759
F1-score: 0.8742

Micro-averaged Metrics:
Precision: 0.8660
Recall: 0.8660
F1-score: 0.8660
Confusion Matrix for Fold 3:
[[328   0   7 109]
 [  1 379   2   2]
 [  2   0 336   4]
 [ 75   0  11 334]]
```

Fold-4

```
Validation Loss:  0.27809264431416525
Early stopping at epoch 6 due to no improvement in validation loss.

Validation Metrics for Fold 4:

Metrics:
Accuracy: 91.41%

Macro-averaged Metrics:
Precision: 0.9284
Recall: 0.9278
F1-score: 0.9280

Micro-averaged Metrics:
Precision: 0.9141
Recall: 0.9141
F1-score: 0.9141
Confusion Matrix for Fold 4:
[[466   0   0  62]
 [  0 300   0   0]
 [  0   0 324   0]
 [ 73   1   0 358]]
```

Fold-5

```
Validation Loss:  0.3767741499235854
Early stopping at epoch 7 due to no improvement in validation loss.

Validation Metrics for Fold 5:

Metrics:
Accuracy: 88.69%

Macro-averaged Metrics:
Precision: 0.9039
Recall: 0.8961
F1-score: 0.8951

Micro-averaged Metrics:
Precision: 0.8869
Recall: 0.8869
F1-score: 0.8869
Confusion Matrix for Fold 5:
[[346   0   0 158]
 [  0 419   0   1]
 [  0   0 419   1]
 [ 49   0   0 455]]
```

Fold-6

```
Validation Loss:  0.7868538759994408
Early stopping at epoch 8 due to no improvement in validation loss.

Validation Metrics for Fold 6:

Metrics:
Accuracy: 90.91%

Macro-averaged Metrics:
Precision: 0.9126
Recall: 0.9142
F1-score: 0.9131

Micro-averaged Metrics:
Precision: 0.9091
Recall: 0.9091
F1-score: 0.9091
Confusion Matrix for Fold 6:
[[431   0   2  79]
 [  0 568   0   0]
 [  0   0 432   0]
 [109   1   1 489]]
```

Fold-7

```
Validation Loss:  0.24360175132751466
Early stopping at epoch 7 due to no improvement in validation loss.

Validation Metrics for Fold 7:

Metrics:
Accuracy: 91.07%

Macro-averaged Metrics:
Precision: 0.9297
Recall: 0.9286
F1-score: 0.9291

Micro-averaged Metrics:
Precision: 0.9107
Recall: 0.9107
F1-score: 0.9107
Confusion Matrix for Fold 7:
[[476   1   0  83]
 [  0 336   0   0]
 [  1   0 325   3]
 [ 76   1   0 546]]
```

Fold-8

```
Validation Loss:  20.175140534341335
Early stopping at epoch 7 due to no improvement in validation loss.

Validation Metrics for Fold 8:

Metrics:
Accuracy: 80.03%

Macro-averaged Metrics:
Precision: 0.8518
Recall: 0.8153
F1-score: 0.8266

Micro-averaged Metrics:
Precision: 0.8003
Recall: 0.8003
F1-score: 0.8003
Confusion Matrix for Fold 8:
[[434   1   6 182]
 [  2 336   0  40]
 [  5   0 265  45]
 [ 80   5   3 444]]
```

Fold-9

```
Validation Loss:  0.6837000325320275
Early stopping at epoch 21 due to no improvement in validation loss.

Validation Metrics for Fold 9:

Metrics:
Accuracy: 73.27%

Macro-averaged Metrics:
Precision: 0.7582
Recall: 0.7443
F1-score: 0.7501

Micro-averaged Metrics:
Precision: 0.7327
Recall: 0.7327
F1-score: 0.7327
Confusion Matrix for Fold 9:
[[ 848   18   33  655]
 [  16 1468    0   28]
 [  31    2  846  108]
 [ 553   16   22  900]]
```

Fold-10

```
Validation Accuracy of Main Architecture: 83.33%
Validation Loss:  0.511651449650526
Early stopping at epoch 8 due to no improvement in validation loss.

Validation Metrics for Fold 10:

Metrics:
Accuracy: 83.71%

Macro-averaged Metrics:
Precision: 0.8456
Recall: 0.8468
F1-score: 0.8436

Micro-averaged Metrics:
Precision: 0.8371
Recall: 0.8371
F1-score: 0.8371
Confusion Matrix for Fold 10:
[[370   0   2 116]
 [  2 459   0   3]
 [  0   7 537   0]
 [212   1   1 402]]
```

Average

```
Average Metrics Across All Folds:
Accuracy: 84.50%
Macro Precision: 0.8616
Macro Recall: 0.8573
Macro F1-score: 0.8575
Micro Precision: 0.8450
Micro Recall: 0.8450
Micro F1-score: 0.8450
```

Best confusion Matrix Across all folds

```
Confusion Matrix for All Folds Combined:
Anger     :[715   0   0  34]
Neutral   :[  0 635   0   0]
Engaged   :[  0   0 550   0]
Bored     :[ 28   0   0 721]
```

Fold	Macro			Micro			Accuracy
	P	R	F	P	R	F	
1	0.7297	0.7401	0.7345	0.7138	0.7138	0.7138	71.38%
2	0.7553	0.7655	0.7590	0.7398	0.7398	0.7398	73.98%
3	0.7469	0.7639	0.7539	0.7435	0.7435	0.7435	74.35%
4	0.8101	0.8210	0.8149	0.7910	0.7910	0.7910	79.10%
5	0.8145	0.8211	0.8173	0.8060	0.8060	0.8060	80.60%
6	0.8051	0.8072	0.8060	0.8097	0.8097	0.8097	80.97%
7	0.8393	0.8456	0.8419	0.8172	0.8172	0.8172	81.72%
8	0.8353	0.8406	0.8375	0.8060	0.8060	0.8060	80.60%
9	0.8228	0.8375	0.8281	0.8246	0.8246	0.8246	82.46%
10	0.8496	0.8630	0.8600	0.8582	0.8582	0.8582	85.82%
Average	0.8019	0.8106	0.8053	0.7910	0.7910	0.7910	79.10%

Table:K-Fold validation results for Part II

Fold	Macro			Micro			Accuracy
	P	R	F	P	R	F	
1	0.7351	0.7461	0.7380	0.7354	0.7354	0.7354	73.54%
2	0.8766	0.8778	0.8768	0.8579	0.8579	0.8579	85.79%
3	0.8739	0.8759	0.8742	0.8660	0.8660	0.8660	86.60%
4	0.9284	0.9278	0.9280	0.9141	0.9141	0.9141	91.41%
5	0.9039	0.8961	0.8951	0.8869	0.8869	0.8869	88.69%
6	0.9126	0.9142	0.9131	0.9091	0.9091	0.9091	90.91%
7	0.9297	0.9289	0.9291	0.9107	0.9107	0.9107	91.07%
8	0.8518	0.8153	0.8266	0.8003	0.8003	0.8003	80.03%
9	0.7582	0.7443	0.7501	0.7327	0.7327	0.7327	73.27%
10	0.8456	0.8468	0.8436	0.8371	0.8371	0.8371	83.71%
Average	0.8616	0.8573	0.8575	0.8450	0.8450	0.8450	84.50%

Table: KFold Validtion for Part III Model

Significant observations or trends from the table include:

1. Consistency: The model's performance varies across different folds, indicating some inconsistency. For instance, the accuracy ranges from a low of 73.24% in Fold 9 to a high of 91.90% in Fold 6.
2. Performance Peaks and Troughs: The model performs best in Fold 6 with the highest Precision, Recall, F1-Score, and Accuracy in both Macro and Micro averages. Conversely, Fold 9 shows the lowest performance across all metrics.
3. Macro vs. Micro Averages: The Macro averages show more variation between folds than the Micro averages, which are relatively more consistent. This suggests that the model's performance is more stable on the overall dataset (as represented by Micro averages) than on the individual classes (as represented by Macro averages).
4. Average Performance: The average performance across all folds is reasonably good, with an average accuracy of 84.50%, Macro F1-Score of 85.75%, and Micro F1-Score of 84.50%. This indicates that, on average, the model is reliable.
5. Potential Overfitting or Data Imbalance: The significant variation in performance across folds could be indicative of overfitting to certain folds or an imbalance in the dataset that affects the model's ability to generalize.

The results from the k-fold cross-validation show a notable difference in performance metrics compared to the original train/test split evaluation. Here's a contrast between the two:

Original Train/Test Split Evaluation (Part II)

- Main Model Accuracy: 75.31%
- Variant 1 Accuracy: 63.51%
- Variant 2 Accuracy: 74.81%

K-Fold Cross-Validation Results

- Average Accuracy: 84.50%

The k-fold cross-validation results in a higher average accuracy compared to the original evaluation. This discrepancy can be attributed to several factors:

1. Data Utilization: K-fold cross-validation uses the entire dataset for both training and validation, across different folds. This comprehensive use of data can lead to a more accurate estimation of the model's performance.
2. Variance Reduction: By averaging the results over multiple folds, k-fold cross-validation reduces the variance of the performance estimate. The original train/test split provides a performance estimate based on a single partition of the data, which can be more susceptible

to the idiosyncrasies of that particular split.

3. Model Stability: The varying performance across different folds in k-fold cross-validation can indicate the model's stability. If the model performs consistently across all folds, it suggests that the model is stable and generalizes well. In contrast, a single train/test split does not provide this level of insight.
4. Bias Correction: The original train/test split might have a bias if the split does not represent the population well. K-fold cross-validation mitigates this by ensuring that each data point is used for validation exactly once, providing a more balanced evaluation.

The k-fold cross-validation results are likely to be more reliable than the original train/test split evaluation due to better data utilization, reduced variance, insights into model stability, and correction for potential bias in the data split.

Bias Analysis:

Data bias, or dataset bias, denotes the existence of systematic and unjust differences within a dataset, which can result in biased outcomes during the training of machine learning models. This bias within the data has the potential to impact the model's predictions and decisions, potentially leading to unjust treatment of various groups.

For our AI project we have considered Age and Gender attributes to test bias analysis. For Age attribute, three groups have been considered such as Young, Middle Aged, Elder. For Gender attribute, two groups have been considered such as Male and Female. We have classified our datasets according to these groups for Anger, Neutral, Engaged and Bored. After classifying each group is tested individually on the saved Model from Part II. For gender category the accuracy was same for both Male and Female. For Age category the accuracy was same for Middle Aged and Senior. But for Young group the accuracy was less compared to other two. So our model is biased to Middle and Elder groups.

Steps for Mitigating Bias:

Outline the measures implemented to mitigate bias, which may encompass modifications to the dataset, model architecture, or training process. Common mitigation steps include:

- Dataset Balancing: Ensure the dataset is reflective of the entire population across various demographic groups.
- Weight Adjustments: Assign different weights to samples from under-represented groups during training to accord them greater significance.

- Regularization Techniques: Employ regularization methods to prevent the model from overly specializing in certain groups and avoid overfitting.
- Sensitive Attribute Removal: If applicable, contemplate removing sensitive attributes from the input data that could contribute to biased predictions.
- Model Retraining: Train the model on the adjusted dataset, incorporating identified mitigation techniques.

To eliminate the bias some extra images for Young has been added for Engaged and Bored dataset as there are less images in these two compared to Anger and neutral. After adding the extra images, the model was trained again and tested with for all the groups. But the accuracies were not as expected. Now we removed some images of Middle and Elder groups to make it even with Young group. Finally, we got the same accuracy for all the groups for Gender attribute.

Values for all groups on the model generated from Part II .

Attribute	Group	Accuracy	Precision	Recall	F1-Score
Age	Young	81.34	0.8452	0.8553	0.8577
	Middle	82.55	0.8553	0.8601	0.8554
	Elder	82.20	0.7872	0.7833	0.7776
	Average	82.03	0.8292	0.8329	0.8302
Gender	Male	82.91	0.8495	0.8515	0.8472
	Female	83.33	0.8314	0.8269	0.8278
	Average	83.12	0.8404	0.8392	0.8375
Average		82.575	0.8348	0.8360	0.8338

Values for all groups on the model generated from Part III.

Attribute	Group	Accuracy	Precision	Recall	F1-Score
Age	Young	91.34	0.9252	0.9064	0.9076
	Middle	92.84	0.9235	0.9240	0.9154
	Elder	90.58	0.9364	0.8977	0.9069
	Average	91.58	0.9283	0.9093	0.9099
Gender	Male	89.98	0.9147	0.8864	0.8903
	Female	92.71	0.9358	0.9264	0.9250
	Average	91.345	0.9252	0.9064	0.9076
Average		91.46	0.9267	0.9078	0.9087

References:

1. https://www.tableau.com/learn/articles/what-is-data-cleaning#:~:text=Data%20cleaning%20is%20the%20process,to%20be%20duplicated%20or%20mislabeled.