COMP 6721

Applied Artificial Intelligence

Fall 2023

Project: A.I.ducation Analytics

Guided By

Professor: Dr. René Witte

TA: Amin Karimi and Naghmeh Shafiee

Group- NS_08

Parth Shah- 40196521(Data Specialist)

Mir Pasad- 40253287 (Training Specialist)

Ankush Desai- 40271170 (Evaluation Specialist)

Github- https://github.com/parthshah312/COMP-6721-Project.git

Table of Contents

Chapter 1: Dataset

- ## Overview of Existing Dataset

Total Number of Images: 2084

Numbers per Class:

Angry- 507

Bored- 501

Focused- 500

Neutral- 576

★ All the images in the dataset are mostly frontal face shots.

- ## Justification for dataset choices

The collection is unusual in that it includes conversational expressions, which are frequently disregarded in facial expression studies. The expressions are created using a method-acting process based on ordinary circumstances, which ensures both well-defined and natural face expressions.

The dataset gains credibility by the application of a method-acting procedure.

The dataset records more realistic and contextually relevant face reactions by connecting expressions to commonplace situations.

The dataset's adaptability is increased by the availability of expressions in various repetitions, intensities, and camera angles.

The database helps researchers in a variety of domains, such as perceptual and cognitive sciences, emotional computing, and computer vision.

Challenges that may be encountered include:

Subjectivity in Expression Interpretation: Assessing and validating face expressions can be subjective, especially in a dataset emphasizing naturalness. It may be difficult to get consensus among human annotators or to construct automated systems capable of precisely comprehending the intended statements.

Data Volume and Processing: Including dynamic expressions and varying repetitions, intensities, and camera angles may result in an increase in overall data volume. Managing and analyzing such a large dataset might be difficult, especially if computer resources are restricted.

Generalization Across Individuals: The dataset contains expressions from 19 different people. It may be necessary to ensure that findings generalize over a larger population. Individual variances in face expressions may have an influence on the project's capacity to draw general findings.

- **Provenance Information**

The whole dataset is uploaded on our project GitHub repository and the images were mainly obtained from two different sources. [2], [3].

Chapter 2: Data Cleaning

- **Techniques**
    - For imperfect size images we resized images into one common sized image (48 x 48).
    - For light augmentation techniques we simply performed brightness enhancing and contrast enhancing on images, by using these techniques we enhanced our images to 50% more brighter.
    - For Reduced Dimensionality , by simplicity and Feature Extraction we simply converted every image into gray scale images.

- **Challenges:**

1. Varied Image Sizes and Resolutions:

Challenge: Images in the dataset had different sizes and resolutions.
Solution: Standardizing the dataset by resizing images to a common dimension.

2. Inconsistent Lighting Conditions:

Challenge: Some images had diverse lighting conditions, affecting the visibility of facial features.
Solution: Brightness adjustments, contrast normalization.

3. Noisy or Mislabeled Data:

Challenge: Some images in the dataset were mislabeled.
Solution: Manually reviewing and correcting mislabeled images.

4. Class Imbalance:

Challenge: The number of images per class were slightly uneven.
Solution: Added more images for that particular class

5. Limited Diversity in Expressions:

Challenge: The dataset lacks diversity in facial expressions, particularly for specific classes.

Solution: Actively seeking and including images that represent a wide range of expressions for each class.

Two attributes were used to segment the data for the bias analysis: gender and age. There are age-related subsegments within both segments: young, middle-aged, and old. There are two genders: male and female. These data were manually labeled by us using visual inspection of the pictures. To make it simpler to load data and parse labels, we have created a hierarchical folder structure with the right class names and dataset segments. To make it more widely applicable, we also used the preprocessing techniques previously discussed, such as contrast, brightness corrections, and augmentation.

The dataset's hierarchy and labeling are displayed in the structure below.

Chapter 3: Labeling

- **Methods And Platforms:**

  First we carefully analyze each and every image of the dataset and put each and every image manually into different types of emotion classes.

  For data labeling, we first load each image with appropriate emotion labels and store it into a labeled list.

  One-hot Encoding: Using this method, we converted each labeled data into binary vectors.

  For example: in our project we detect 4 type of emotions (angry, bored, neutral, bored) so, by one hot encoding these labeled class look like,

  Angry [1,0,0,0]

  Bored [0,1,0,0]

  Neutral [0,0,1,0]

  Bored [0,0,0,1]

Challenges faced when merging dataset:

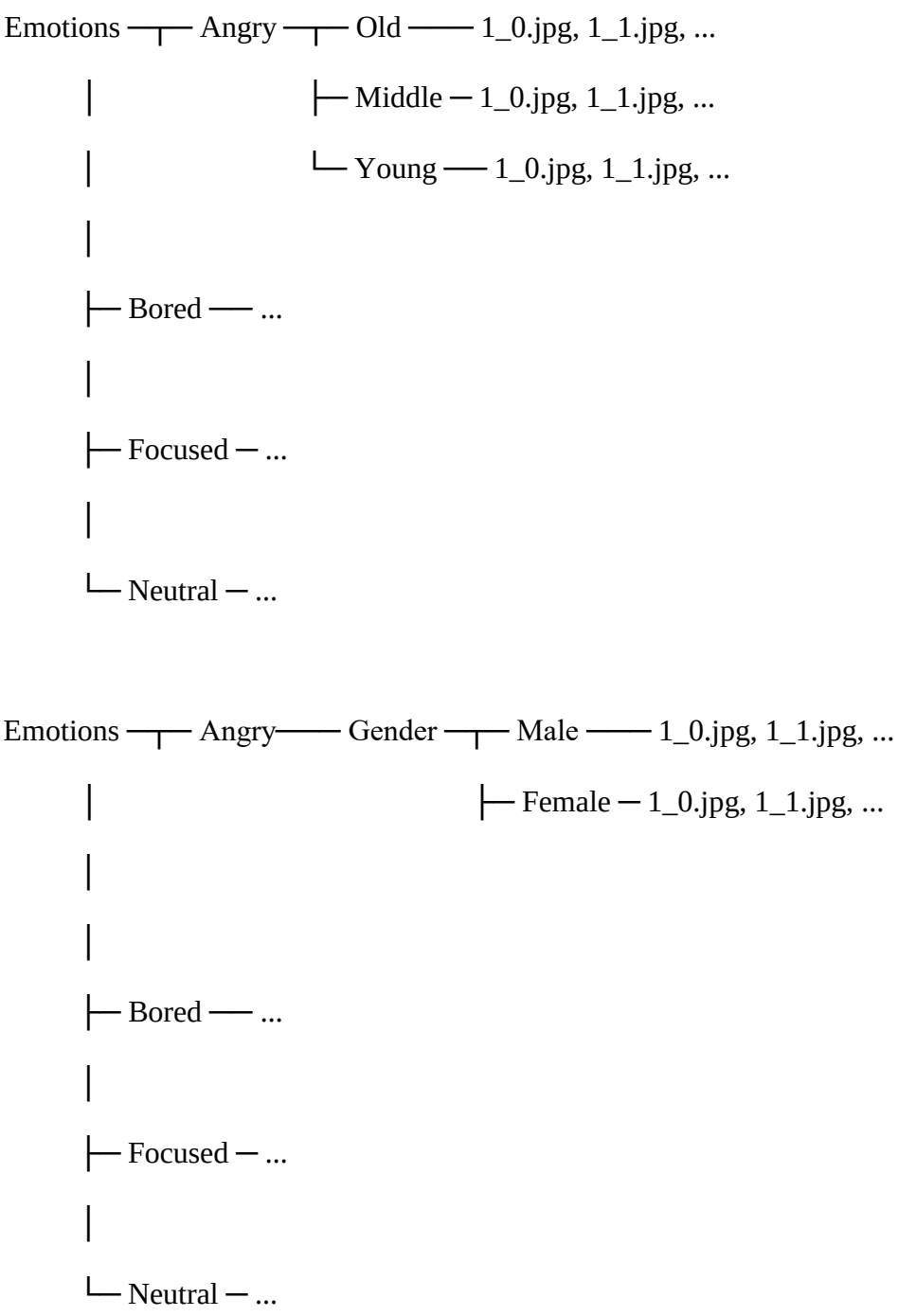
1. Label Consistency:

   Challenge: Different naming conventions were used across different dataset.
   Solution: We manually adjust labels to make them consistent.

2. Data Quality:

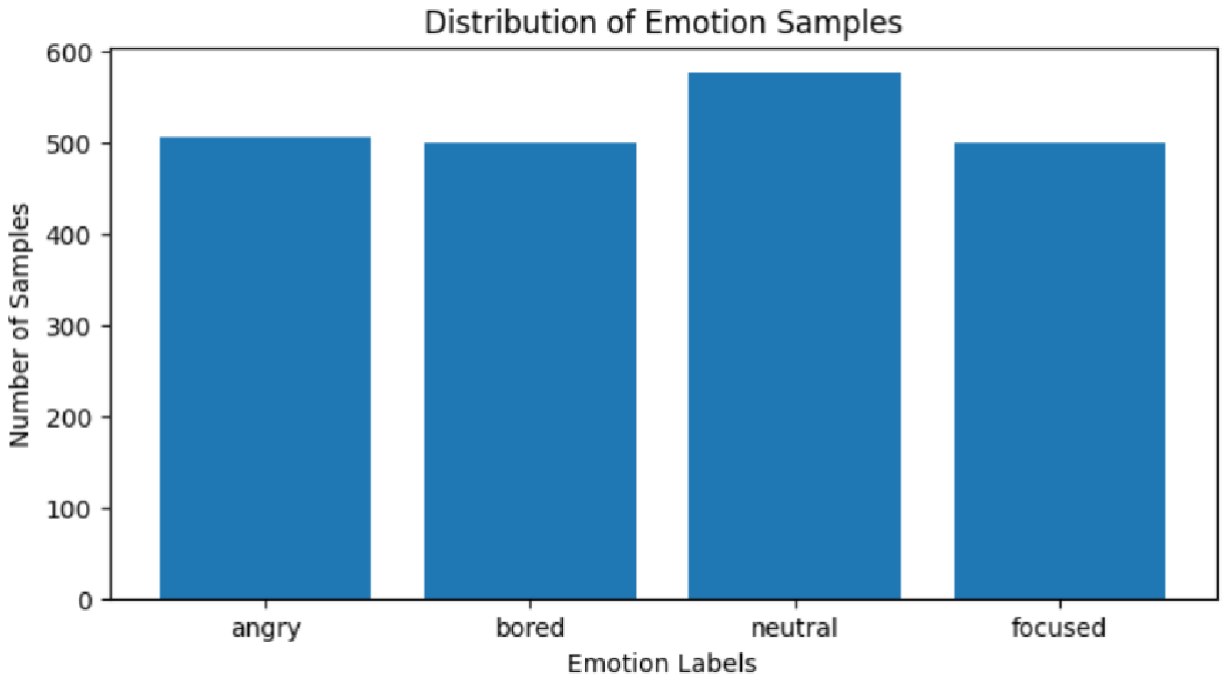
   Challenge: Datasets had varying levels of image quality.
   Solution: We removed low quality images.

We labeled the data structure as below; we chose the Age and Gender for our bias analysis.



Chapter 4: Data Visualization

- **Class Distribution:**



Bar Graph: A bar graph is used to visualize the count of images in each class. Each bar represents a facial expression class, and the height of the bar corresponds to the number of images in that class.

Labels: Each bar is labeled with the corresponding class name for clarity.

Imbalance Check: Analyze if certain classes are overrepresented or underrepresented. A balanced distribution helps in training machine learning models more effectively.

We have displayed a bar graph showing the number of images for each class. Labeled each bar with the corresponding class name. We have also analyzed for a nearly equal distribution of classes.

- **Sample Images:** Provide a visual representation of the dataset's content through a set of randomly selected images.

  Purpose: Identify anomalies, ensure diversity in samples, and visually inspect the variety of facial expressions within each class.

  Grid Layout: A 5x5 grid is used to display a collection of 25 images, with each row

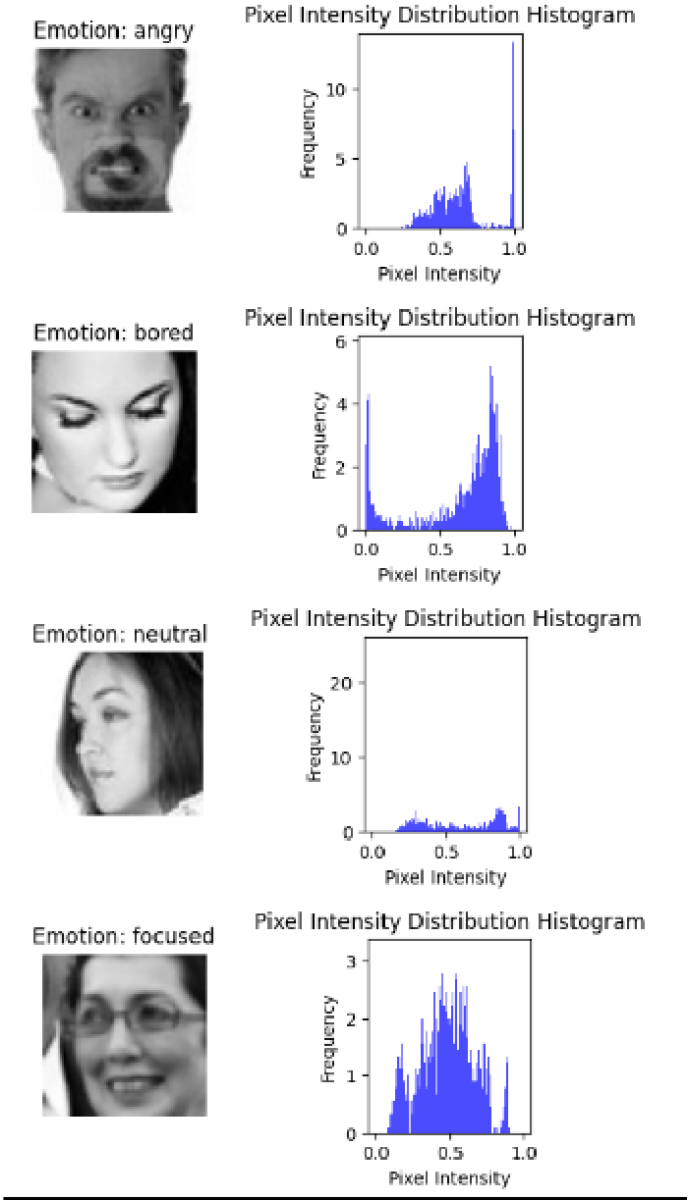
representing a different class.

Random Selection: Images are randomly chosen from each class to ensure a representative sample.

Each image is labeled with its corresponding class to aid interpretation.

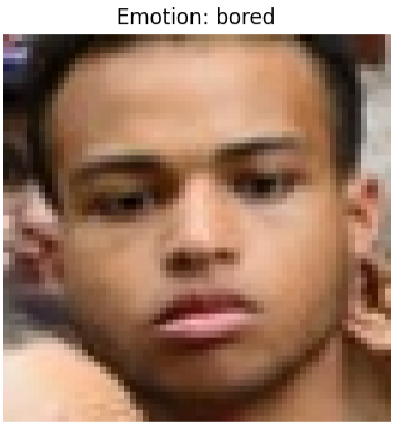
Pixel Intensity Distribution: Understand the distribution of pixel intensities in images, providing insights into lighting conditions. Assess potential challenges related to image quality, such as variations in lighting conditions.

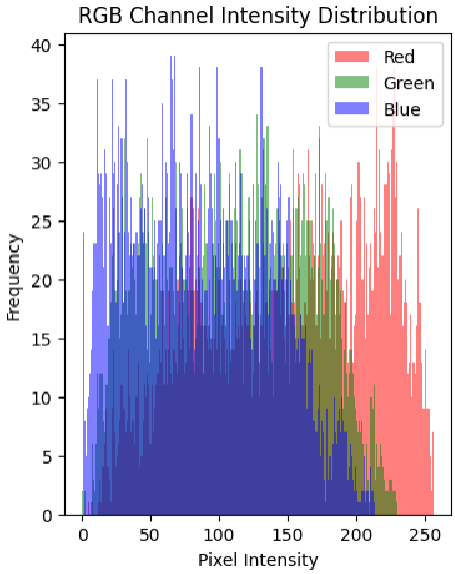
Histogram: A histogram is plotted to visualize the distribution of pixel intensities in a set of random images.



- **<u>RGB colored image intensity Distribution:</u>** Working with color images, overlay histograms for the Red, Green, and Blue channels on a single plot. This provides insights into color variations.

  Frequency vs. Intensity: The x-axis represents pixel intensity, and the y-axis represents the frequency of pixels at each intensity level.





Chapter 5: CNN Architecture

1. Model Overview and Architecture Details:

When it comes to "A.I.ducation Analytics," a strong Convolutional Neural Network (CNN) is essential. To maximize its performance, our primary model is built with meticulous consideration for architectural elements and subtle design details. Let's examine our CNN architecture in more detail, including its layers, activation functions, and special features. We'll also look at the two versions that were added during the experimental stage.

Activation functions and layers:

Three convolutional layers and three fully linked layers make up our primary model. A 3 x 3 kernel size with a padding of 1 and a stride of 1 is used in each convolutional layer. Leaky ReLU is the activation function that we have selected for our model; it introduces non-linearity and permits the propagation of modest negative values.

Layer 1:

The first layer is the Convolution layer with kernel size 3 X 3 and padding 1. It produces 1 input channel and output with 64 channels.

Layer 2:

Convolution layer with 64 input channels, 128 output channels and kernel size 3 X 3 with padding 1.

Layer 3:

Convolution layer with 128 input channels, 256 output channels and kernel size 3 X 3 with padding 1.

Architectural Components:

Two dropout layers are positioned thoughtfully in our architecture to serve as a regularization strategy during training and prevent overfitting. To down sample the spatial dimensions and lower computational complexity while maintaining critical information, MaxPooling layers are integrated as pooling layers. We have used MaxPooling with Kernel size 2 and stride 2.

Fully Connected Layer 1:

The Hidden Layer contains 9216 neurons, and output layer contains 1024.

Fully Connected Layer 2:

The Hidden Layer contains 1024 neurons, and output layer contains 512.

Fully Connected Layer 3:

The Hidden Layer contains 512 neurons, and output layer contains 4.

Design Nuances:

We took great care to balance efficiency and complexity in our model's architecture. Convolutional and fully linked layers are used in this way to allow for the learning of hierarchical features, ranging from simple edges and textures to intricate facial structures. The Leaky ReLU activation function enhances the flexibility of the model's feature extraction procedure by allowing it to accommodate negative values.

2. **Variant 1:**

VGG19 Architecture: We chose the well-known VGG19 architecture for our initial variant. With three fully connected layers and five convolutional layers, this version presents a deeper network. The use of VGG19 is intended to investigate how deeper water affects the model's realism in capturing fine-grained facial characteristics. This form adds two dropout layers and preserves the ReLU activation function, providing a deeper comprehension of face features.

Convolution Layers:

Layer 1:

The first layer is the Convolution layer with kernel size 3 with padding 1. It produces 1 input channel and output with 64 channels.

Layer 2:

Convolution layer with 64 input channels, 128 output channels and kernel size 3 with padding 1.

Layer 3:

Convolution layer with 128 input channels, 256 output channels and kernel size 3 with padding 1.

Layer 4:

Convolution layer with 256 input channels, 512 output channels and kernel size 3 with padding 1.

Layer 5:

Convolution layer with 512 input channels, 512 output channels and kernel size 3 with padding 1.

Fully Connected Layer 1:

The Hidden Layer contains 25088 neurons, and output layer contains 4096.

Fully Connected Layer 2:

The Hidden Layer contains 4096 neurons, and output layer contains 4096.

Fully Connected Layer 3:

The Hidden Layer contains 4096 neurons, and output layer contains 4.

3. Variant 2:

Expanded Kernel Size: In this version, we extended the kernel size by choosing a 5 x 5 arrangement. The goal of this architectural modification is to evaluate the trade-offs between computational expense and geographical granularity. The model is anticipated to prioritize the recognition of larger facial features over finer details by increasing the kernel size. At the evaluation stage, a thorough analysis of the influence of this modification on the model's recognition capabilities will be conducted.

Convolutional Layers:

Layer 1:

The first layer is the Convolution layer with kernel size 5 X 5 and padding 1. It produces 1 input channel and output with 64 channels.

Layer 2:

Convolution layer with 64 input channels, 128 output channels and kernel size 5 X 5 with padding 1.

Layer 3:

Convolution layer with 128 input channels, 256 output channels and kernel size 5 X 5 with padding 1.

Fully Connected Layer 1:

The Hidden Layer contains 1024 neurons, and output layer contains 1024.

Fully Connected Layer 2:

The Hidden Layer contains 1024 neurons, and output layer contains 512.

Fully Connected Layer 3:

The Hidden Layer contains 512 neurons, and output layer contains 4.

4. Training Process:

Throughout 80 epochs, a learning rate of 0.00001 was used to train the models. We have chosen Cross-entropy loss as the loss function, and Adam is an optimizer. To train the model, we tried a range of kernel sizes. Three and five different kernel sizes have been employed. The model performed best when a 3-layer convolution net was used with a 3x3 kernel.

Results:

Main model 3x3 Kernel:

Following 80 epochs, the model with the stated kernel size attained a training accuracy of around 84.36%, a testing accuracy of 65.71%, and a validation accuracy of 66.13%. When compared to the other kernel, it performed better.

```
Epoch [65/80], Tr Loss: 0.5527, Tr Acc: 78.6694, Val Loss: 0.864637, Val Acc: 65.495209
Epoch [66/80], Tr Loss: 0.5491, Tr Acc: 77.1605, Val Loss: 0.875890, Val Acc: 64.217255
Epoch [67/80], Tr Loss: 0.5510, Tr Acc: 76.5432, Val Loss: 0.846920, Val Acc: 66.453674
Epoch [68/80], Tr Loss: 0.5224, Tr Acc: 78.3265, Val Loss: 0.852822, Val Acc: 65.495209
Epoch [69/80], Tr Loss: 0.5163, Tr Acc: 78.3265, Val Loss: 0.833690, Val Acc: 65.175720
Epoch [70/80], Tr Loss: 0.4942, Tr Acc: 80.1097, Val Loss: 0.850402, Val Acc: 63.897766
Epoch [71/80], Tr Loss: 0.5006, Tr Acc: 80.1097, Val Loss: 0.849136, Val Acc: 66.773163
Epoch [72/80], Tr Loss: 0.4843, Tr Acc: 80.6584, Val Loss: 0.845640, Val Acc: 64.217255
Epoch [73/80], Tr Loss: 0.4817, Tr Acc: 81.3443, Val Loss: 0.858645, Val Acc: 65.814697
Epoch [74/80], Tr Loss: 0.4425, Tr Acc: 82.5789, Val Loss: 0.904472, Val Acc: 63.897766
Epoch [75/80], Tr Loss: 0.4599, Tr Acc: 80.8642, Val Loss: 0.897322, Val Acc: 66.134186
Epoch [76/80], Tr Loss: 0.4542, Tr Acc: 81.3443, Val Loss: 0.864969, Val Acc: 66.773163
Epoch [77/80], Tr Loss: 0.4544, Tr Acc: 82.1674, Val Loss: 0.921109, Val Acc: 65.495209
Epoch [78/80], Tr Loss: 0.4313, Tr Acc: 82.7846, Val Loss: 0.917887, Val Acc: 67.412140
Epoch [79/80], Tr Loss: 0.4302, Tr Acc: 83.3333, Val Loss: 0.906059, Val Acc: 64.856232
Epoch [80/80], Tr Loss: 0.4008, Tr Acc: 84.3621, Val Loss: 0.948464, Val Acc: 66.134186
```

Variant 1:

The models change 3 layers convolution to 5 layers, under training for 100 epochs using a learning rate of 0.00001 and a kernel size of 3. We have chosen Cross-entropy loss as the loss function, while Adam is an optimizer. The model achieved a training accuracy of around 79.90% and a testing accuracy of 60.70% and a validation accuracy of 57.50 %.

```
Epoch [85/100], Tr Loss: 0.5528, Tr Acc: 76.2003, Val Loss: 1.114243, Val Acc: 61.022366
Epoch [86/100], Tr Loss: 0.5386, Tr Acc: 76.6804, Val Loss: 1.043532, Val Acc: 61.022366
Epoch [87/100], Tr Loss: 0.5337, Tr Acc: 77.0919, Val Loss: 0.975005, Val Acc: 61.980831
Epoch [88/100], Tr Loss: 0.5226, Tr Acc: 77.9150, Val Loss: 1.063114, Val Acc: 61.661339
Epoch [89/100], Tr Loss: 0.5470, Tr Acc: 76.2689, Val Loss: 0.993515, Val Acc: 58.466454
Epoch [90/100], Tr Loss: 0.5695, Tr Acc: 75.3772, Val Loss: 0.944017, Val Acc: 60.702873
Epoch [91/100], Tr Loss: 0.5241, Tr Acc: 76.6804, Val Loss: 1.156326, Val Acc: 61.022366
Epoch [92/100], Tr Loss: 0.5048, Tr Acc: 78.4636, Val Loss: 1.080122, Val Acc: 62.619804
Epoch [93/100], Tr Loss: 0.5649, Tr Acc: 75.9259, Val Loss: 1.030202, Val Acc: 59.424919
Epoch [94/100], Tr Loss: 0.5310, Tr Acc: 76.9547, Val Loss: 1.085662, Val Acc: 62.300320
Epoch [95/100], Tr Loss: 0.5142, Tr Acc: 77.2977, Val Loss: 1.031995, Val Acc: 61.022366
Epoch [96/100], Tr Loss: 0.4915, Tr Acc: 79.6982, Val Loss: 1.133656, Val Acc: 62.300320
Epoch [97/100], Tr Loss: 0.5278, Tr Acc: 77.7778, Val Loss: 1.025365, Val Acc: 62.300320
Epoch [98/100], Tr Loss: 0.4802, Tr Acc: 79.1495, Val Loss: 1.236598, Val Acc: 61.022366
Epoch [99/100], Tr Loss: 0.4588, Tr Acc: 81.0014, Val Loss: 1.204501, Val Acc: 62.619804
Epoch [100/100], Tr Loss: 0.4830, Tr Acc: 79.9040, Val Loss: 1.224823, Val Acc: 57.507984
```

Variant 2:

The models changed 3 kernel size to 5 kernel size, underwent training for 100 epochs using a learning rate of 0.00001 and a kernel size of 5. We have chosen Cross-entropy loss as the loss function, while Adam is an optimizer. The model achieved a training accuracy of around 88.75% and a testing accuracy of 63.89% and a validation accuracy of 61.98%.

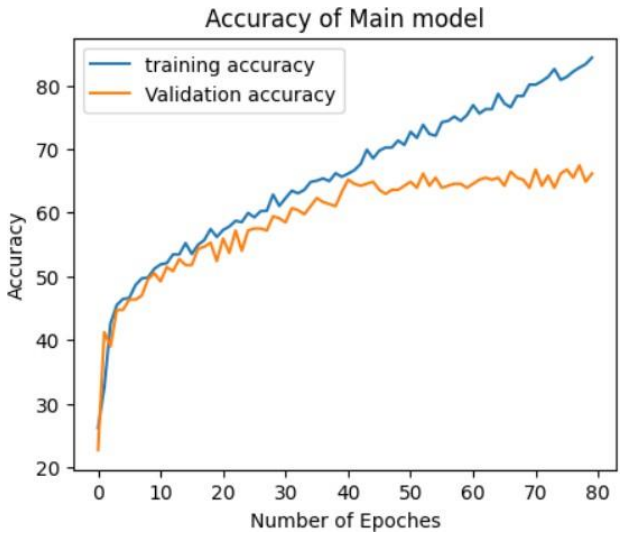
```
Epoch [85/100], Tr Loss: 0.4673, Tr Acc: 79.7668, Val Loss: 1.122540, Val Acc: 60.383385
Epoch [86/100], Tr Loss: 0.4155, Tr Acc: 82.6475, Val Loss: 1.020994, Val Acc: 60.702873
Epoch [87/100], Tr Loss: 0.4157, Tr Acc: 82.7160, Val Loss: 1.114942, Val Acc: 61.980831
Epoch [88/100], Tr Loss: 0.3933, Tr Acc: 84.6365, Val Loss: 1.233695, Val Acc: 59.105431
Epoch [89/100], Tr Loss: 0.3860, Tr Acc: 84.7051, Val Loss: 1.120575, Val Acc: 61.341850
Epoch [90/100], Tr Loss: 0.3847, Tr Acc: 84.3621, Val Loss: 1.157611, Val Acc: 62.300320
Epoch [91/100], Tr Loss: 0.3878, Tr Acc: 84.5679, Val Loss: 1.154385, Val Acc: 61.980831
Epoch [92/100], Tr Loss: 0.3631, Tr Acc: 85.7339, Val Loss: 1.161813, Val Acc: 59.105431
Epoch [93/100], Tr Loss: 0.3734, Tr Acc: 85.0480, Val Loss: 1.164812, Val Acc: 61.980831
Epoch [94/100], Tr Loss: 0.3362, Tr Acc: 86.2826, Val Loss: 1.267612, Val Acc: 59.744408
Epoch [95/100], Tr Loss: 0.3447, Tr Acc: 86.4883, Val Loss: 1.271942, Val Acc: 61.980831
Epoch [96/100], Tr Loss: 0.3470, Tr Acc: 85.3909, Val Loss: 1.162644, Val Acc: 61.980831
Epoch [97/100], Tr Loss: 0.3175, Tr Acc: 87.6543, Val Loss: 1.315669, Val Acc: 60.702873
Epoch [98/100], Tr Loss: 0.3192, Tr Acc: 87.5857, Val Loss: 1.268382, Val Acc: 63.258785
Epoch [99/100], Tr Loss: 0.2929, Tr Acc: 88.4774, Val Loss: 1.270722, Val Acc: 61.022366
Epoch [100/100], Tr Loss: 0.2944, Tr Acc: 88.7517, Val Loss: 1.294727, Val Acc: 61.980831
```

Chapter 6: Evaluation

Visualizing Model's Training and Validation Accuracy and Loss:

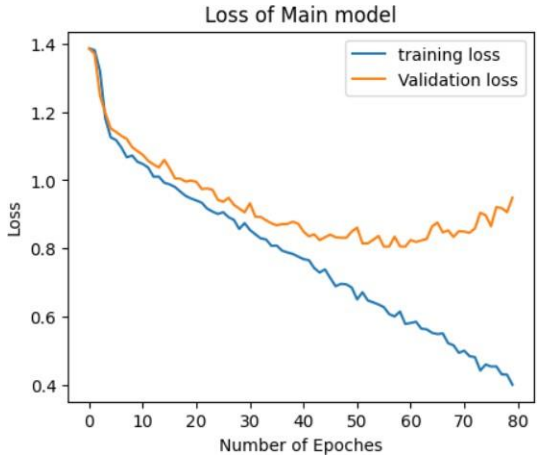
Main Model:

Visualizing Accuracy:



From the above graph we can say that when the number of epochs increases, the accuracy of main model also increases.

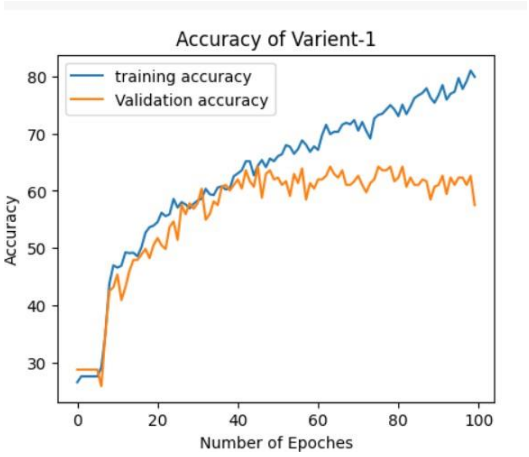
Visualizing Loss:



From the above graph we can say that when the number of epochs increases, the loss of the main model decreases.

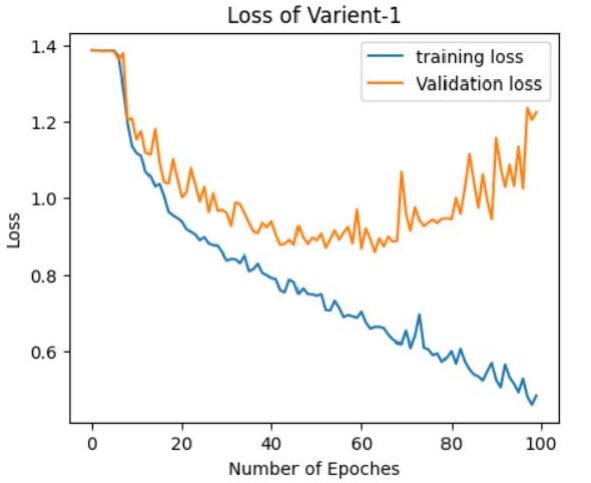
Variant 1:

Visualizing Accuracy:



From the above graph we can say that when number of epochs increases, the training accuracy of variant 1 model also increases but at the same time for the validation accuracy remains constant after 60%. So, this model performs overfitting.

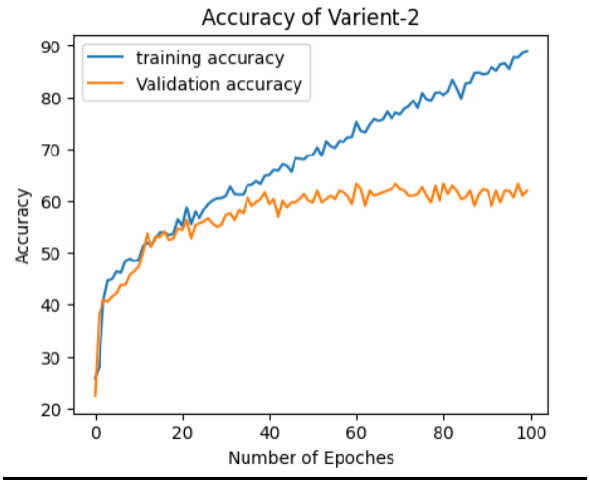
Visualizing Loss:



From the above graph we can say that when number of epochs increases, the training loss of variant 1 model decreases but at the same time for the validation loss after 60 epochs it starts increasing. So, this model performs overfitting.

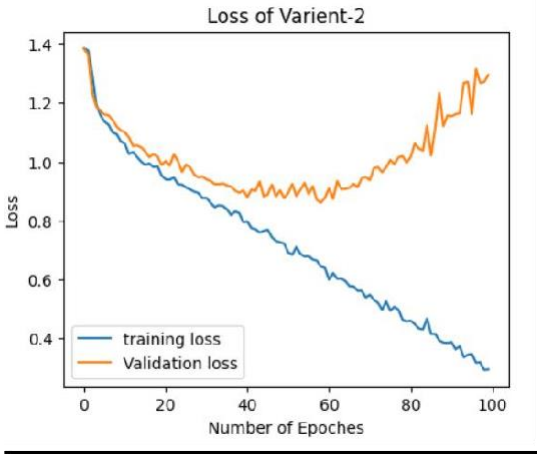
Variant 2:

Visualizing Accuracy:



From the above graph we can say that when number of epochs increases, the training accuracy of variant 2 model also increases but at the same time for the validation accuracy remains constant after 60%. So, this model performs overfitting.

Visualizing Loss:



From the above graph we can say that when number of epochs increases, the training loss of variant 2 model decreases but at the same time for the validation loss after 60 epochs it starts increasing. So, this model performs overfitting.

Performance Metrics:

Main Model:

```
Evaluation of Main model on Testing Data :
Precision_macro: 0.6695
precision_micro: 0.6581
Recall_macro: 0.6659
recall_micro: 0.6581
Accuracy: 0.6581
F1-Measure_macro: 0.6573
F1-Measure_micro: 0.6581
```

Variant 1:

```
Evaluation of Varient-1 on Testing Data :
Precision_macro: 0.6228
precision_micro: 0.6070
Recall_macro: 0.6173
recall_micro: 0.6070
Accuracy: 0.6070
F1-Measure_macro: 0.6182
F1-Measure_micro: 0.6070
```

Variant 2:

```
Evaluation of Varient-2 on Testing Data :
Precision_macro: 0.6448
precision_micro: 0.6390
Recall_macro: 0.6447
recall_micro: 0.6390
Accuracy: 0.6390
F1-Measure_macro: 0.6384
F1-Measure_micro: 0.6390
```

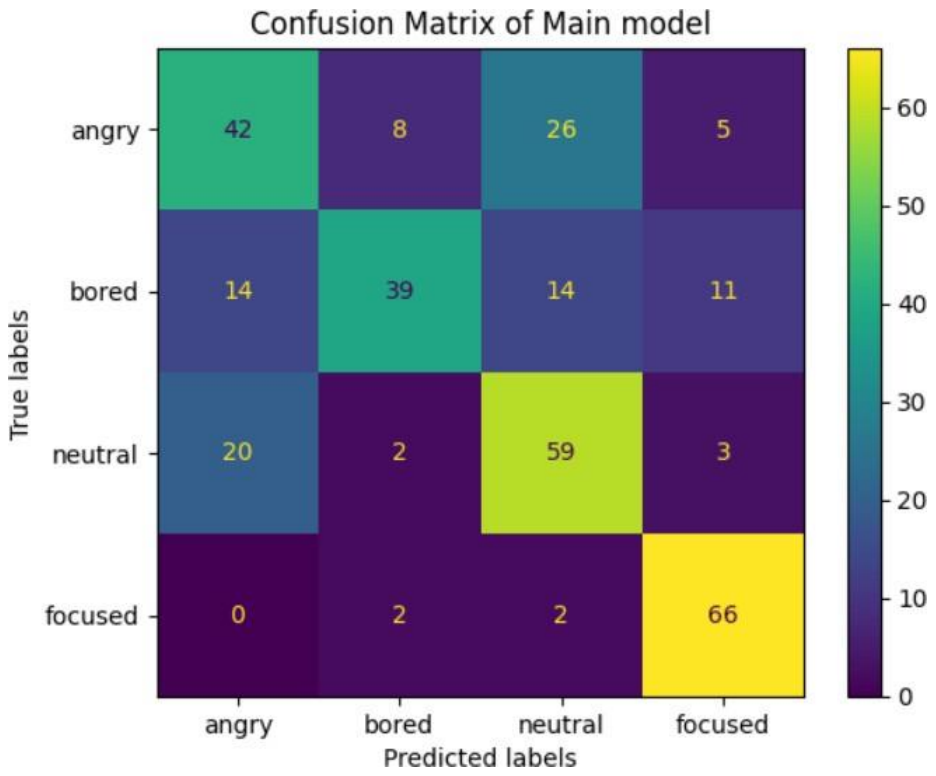
Model	Macro			Micro			Accuracy
	P	R	F1	P	R	F1	
Main Model	0.6695	0.6659	0.6573	0.6581	0.6581	0.6581	0.6581
Variant1	0.6228	0.6173	0.6182	0.6070	0.6070	0.6070	0.6070
Variant2	0.6448	0.6447	0.6384	0.6390	0.6390	0.6390	0.6390

The main model has a testing accuracy that is higher than Variant 1 and Variant 2.

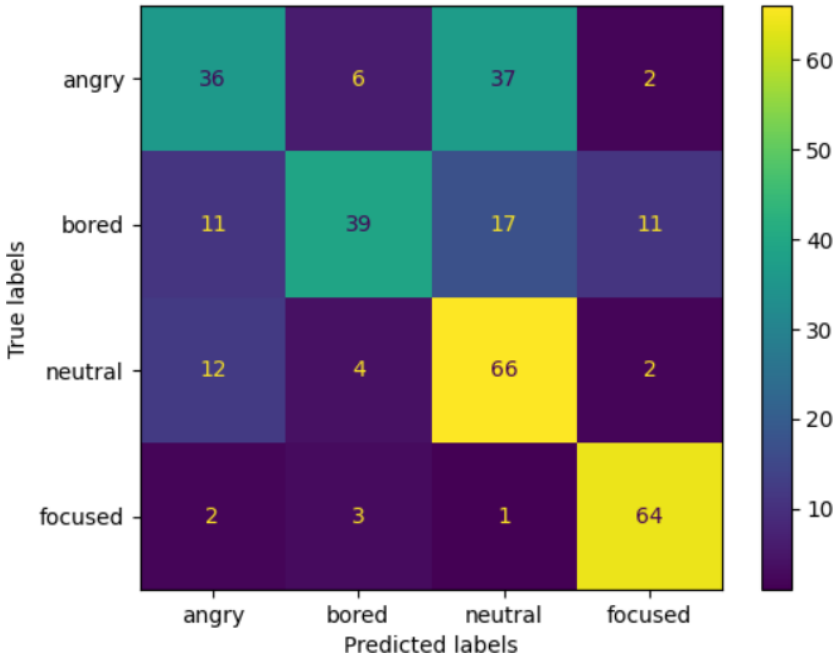
Confusion Matrix Analysis:

The prevalent misclassification in the confusion matrix occurs with the "boring" class, primarily stemming from subtle visual distinctions leading to an elevated number of false negatives. Conversely, the model demonstrates proficiency in accurately identifying neutral and focused facial expressions. These classes exhibit substantial visual disparities, providing distinct feature sets for the model to learn from. In contrast, the more nuanced features of angry and bored expressions contribute to a higher likelihood of misclassifications in these categories. This underscores the model's tendency to exhibit bias towards classes with less distinct visual characteristics.

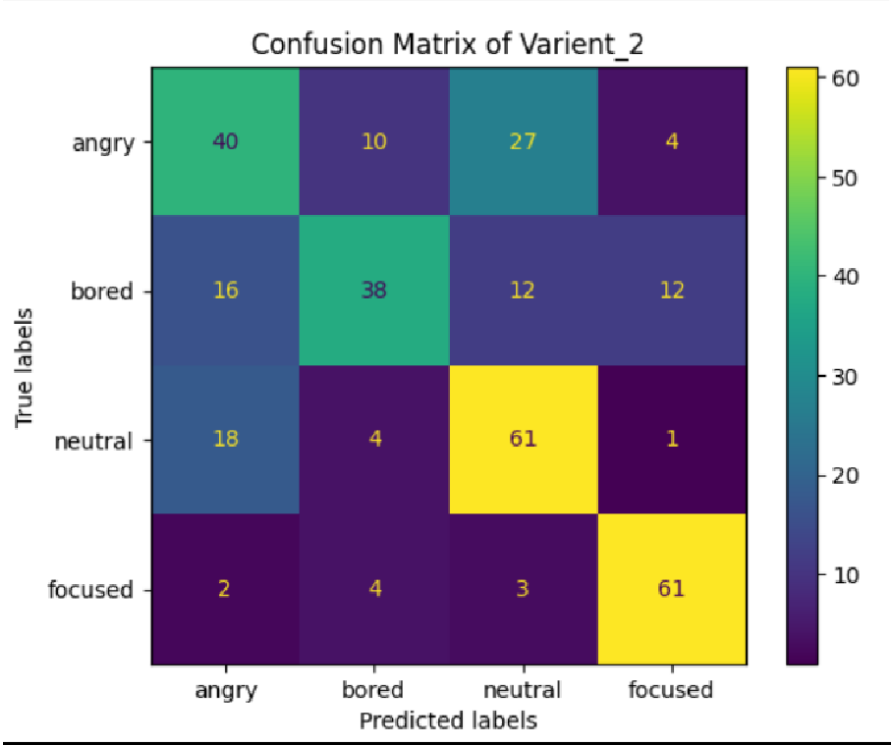
Main Model:



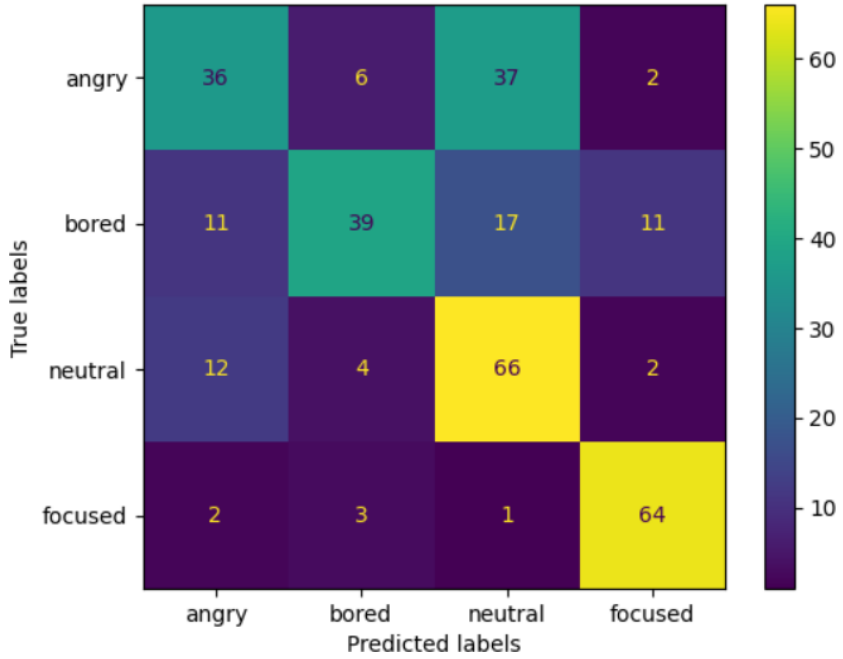
Variant 1:



Variant 2:



Confusion Matrix (Final Model):



Impact of Architectural Variation:

We explored three distinct models, and Variant 1, featuring five convolution layers, encountered difficulties in capturing sufficient details for effective classification. The model grappled with overfitting issues, prompting us to mitigate the problem by reducing the depth of the network. This adjustment involved the removal of two convolution layers to enhance the model's ability to extract intricate features.

In our exploration of model training, we conducted experiments involving varying kernel sizes—specifically, 3 and 5. Notably, the model exhibited optimal performance when utilizing a 3x3 kernel. The detailed insights into these experiments are elaborated upon in the Training Process section above.

Chapter 7: K-Fold Cross Validation & Bias Analysis

K-fold Cross Validation:

The tables below will explain all the results of the k-fold cross-validation for both of our models. To implement K-fold cross-validation we took help of python's scikit learn library [4]:

Fold	Macro			Micro			Accuracy
	P	R	F	P	R	F	
1	70.07	68.33	68.61	70.21	71.20	70.86	70.260
2	70.07	68.33	68.61	70.21	70.21	70.21	70.260
3	68.07	69.33	68.61	70.21	71.20	70.21	70.260
4	68.67	68.02	68.77	70.56	70.98	70.56	71.122
5	69.11	68.33	69.64	70.33	71.03	70.33	70.331
6	68.93	68.96	69.53	70.86	70.08	70.86	70.861
7	68.89	68.06	68.92	69.44	69.03	69.89	69.443
8	69.60	69.93	68.23	70.86	70.86	70.86	70.863
9	69.11	68.33	68.64	70.33	70.41	70.30	70.867
10	69.97	68.91	69.29	70.99	71.30	71.30	70.996
Average	68.49	69.71	69.22	70.66	71.08	70.74	70.567

Table 1: K-fold cross-validation of our final model

Fold	Macro			Micro			Accuracy
	P	R	F	P	R	F	
1	59.69	59.22	59.24	59.36	59.36	59.36	59.36
2	59.93	59.94	59.90	59.33	59.33	59.33	59.33

3	59.81	59.92	59.95	59.64	59.64	59.64	59.64
4	59.97	59.97	59.93	59.01	59.06	59.56	59.56
5	59.83	59.87	59.91	59.85	59.88	59.89	59.89
6	59.93	59.99	59.93	59.91	59.90	59.90	59.90
7	59.90	59.97	59.89	59,92	59.83	59.99	59.95
8	59.11	59.08	59.90	59.12	59.22	59.38	59.38
9	59.02	59.10	59.98	59.10	59.13	59.13	59.13
10	59.04	59.90	59.93	59.83	59.88	59.80	59.80

Table 2: K-fold cross-validation from Part II

The table shows that our final model performed better than the original one. It performs quite well even if the results in Macro precision, recall, and f1-score are nearly identical to those in Micro precision, recall, and accuracy. The nature of the dataset accounts for the variation in performance indicators between the macro and micro averages. While having trouble with minority classes, the model is doing well with the majority class. Our dataset is becoming more balanced as we add more example photos, but it still has trouble correctly classifying in certain minority classes.

There is a noticeable change in the results when we compare the output from our prior model with the output from the k-fold cross-validation. This is due to several factors. The dataset encompasses a greater variety of data variations and is, first, more balanced. Second, the first train-test split was not indicative of the entire dataset, as evidenced by the considerable difference in performance between 10-fold cross-validation and a single train-test split. This problem is mitigated by cross-validation, which uses several train-test splits. in order to ensure that all classes and data variants are fairly represented. As a result, our test accuracy is higher than previously.

Bias Analysis:

Introduction and Result:

For bias analysis, we have chosen age and gender as the attributes to analyze biases throughout different classes.

Gender: We divided the test dataset into groups of Male and Female in order to assess gender bias. We may examine whether the model performs differently depending on gender thanks to this distinction.

Age: We separated the test dataset into three age groups—young, middle-aged, and old—to see how our model functions in each without biasing results. We were able to evaluate the model's consistency throughout all stages thanks to this categorization.

The table below shows Accuracy, Precision, Recall, and F1-measure for both attributes and for both of our models:

Attribute	Group	Accuracy%	Precision%	Recall%	F1-Score%
Gender	Male	64.71	64.56	64.89	64.72
	Female	67.54	67.86	67.20	67.52
Age	Young	64.79	63.92	63.56	63.74
	Middle	62.76	62.25	62.12	62.18
	Old	64.45	64.77	63.08	63.91
Overall System Average		63.18	63.51	64.18	65.61

Table 3: Bias analysis among different attributes using initial model.

Attribute	Group	Accuracy%	Precision%	Recall%	F1-Score%
Gender	Male	64.27	66.56	64.27	65.34
	Female	62.34	61.87	62.34	62.09
Age	Young	61.56	60.20	61.56	60.59
	Middle	65.01	63.61	65.91	65.12
	Old	67.65	68.19	67.65	68.09
Overall System Average		66.37	64.01	66.06	64.81

Table 4: Bias analysis among different attributes using Final model.

Detecting and Mitigating Bias:

Table 4 shows us how well our model works over a range of classes and features. It suggests that our model has good generalization. With an accuracy of 44.27%, the male class performs worse

than the others across the four qualities; the female class performs best, with an accuracy of 52.34%.

But our original attempt at a bias analysis yielded very poor results. While certain qualities yielded very high accuracy, others were almost at zero. Therefore, we carefully tagged our dataset to help alleviate this problem. Another problem was that the online train-test-validation split we had been employing for our training caused some data leakage, which made the performance unacceptable. After properly splitting the data, we retrained the model and obtained these outcomes.

Our bias analysis and the model's overall performance have both improved because of all these bias reducing techniques. A noteworthy accomplishment, as we can see when we examine the total model performance from Tables 3 and 4, is the 7.19% improvement in our model's test accuracy.

Chapter 8: Conclusion

The model with a 3x3 kernel and 3 convolution layers emerged as the most effective, achieving a harmonious balance between training and testing accuracy (84.36% and 65.81%, respectively). This balance indicates successful feature extraction without notable issues of overfitting or underfitting. Larger kernel sizes, such as 5x5, resulted in overfitting. The 3-layer architecture proved optimal for our dataset, with each configuration change significantly influencing performance.

To enhance accuracy in training and validation, consider implementing data augmentation techniques. This involves diversifying the training set by introducing variations like rotation, resizing, or flipping of images. Such approaches contribute to a more comprehensive training set, promoting better generalization by the model. Additionally, explore the utilization of pre-trained models, fine-tuning them on your specific dataset. This technique leverages learned features from extensive and diverse datasets, potentially improving the model's performance on your facial image analysis task.

Using the K-fold method for evaluation, we discovered problems with our model. Thus, we modified our model and data. In order to normalize the data, we defined a transform. To get the greatest fit for our model, we changed the normalization values. Enhancing the model's performance required this. This was a critical step in improving and strengthening our AI.

References

[1] Kaulard K, Cunningham DW, Bülthoff HH, Wallraven C (2012),"The MPI Facial Expression Database — A Validated Database of Emotional and Conversational Facial Expressions," PLoS ONE 7(3): e32321. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0032321

[2] Facial-Expression-Classification-Dataset(3Classes) dataset for Facial Recognition(FER-2013 dataset; accessed October 24, 2023).
https://www.kaggle.com/datasets/nightfury007/fercustomdataset-3classes

[3] The large MPI Facial Expression- A Validated Database of Emotional and Conversational Facial Expressions(MPI dataset; accessed October 25, 2023) https://www.b-tu.de/en/graphic-systems/databases/the-large-mpi-facial-expression-database

[4] *Image Augmentation*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/04/image-augmentation-using-3-python-librarie/

[5] *Cross-validation: Evaluating estimator performance*. Scikit Learn. https://scikit-learn.org/stable/modules/cross_validation.html