COMP 6721

Applied Artificial Intelligence

Fall 2023

Project Assignment Part 1 & 2 – Team AK_8

Name	Specialization	Student ID
Krutik Gevariya	Data Specialist	40232386
Shivam Patel	Training Specialist	40226428
Nayankumar Rajeshbhai Sorathiya	Evaluation Specialist	40227432

GitHub Link : https://github.com/nayansorarhiya/AAI_Project.git

Dataset Information

Overview

In our emotion recognition project, we employed two datasets: "Dataset A" and "Dataset B" to classify emotions into four distinct categories, namely **Neutral**, **Engaged**, **Bored**, and **Angry**.

- **Dataset A:**
  - Total Number of Images: 1,118
  - Number of Images per **Angry**, **Neutral** Class: 559

- **Dataset B:**
  - Total Number of Images: 1,110
  - Number of Images per **Boredom**, **Engagement** Class: 555

Total : 2,228

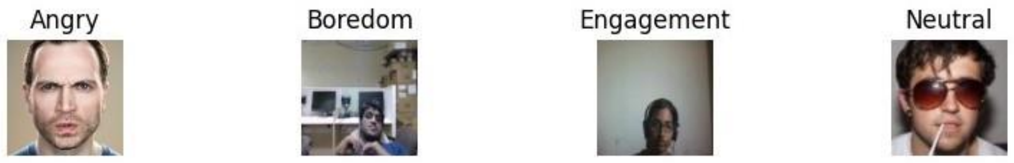


Figure 1. Facial Expression Categories

Attributes : Dataset A predominantly comprises frontal facial images featuring varied lighting conditions, a wide range of ages, gender, and diverse backgrounds.

Justification for Dataset selection

- We opted for **Dataset A** because of the high quantity of images and the concentration of **frontal face shots**, which are beneficial for training a reliable face recognition model.
- **Dataset B** was chosen to add variability to the training data, since it includes photos taken in various contexts. This combination of datasets allows our model to have a successful performance in **real-world scenarios** where lighting and backgrounds may differ. Nonetheless, this also presents a challenge when it comes to managing the augmented diversity of the data.

Provenance Information

Dataset	Image Source	Licensing Type	Relevant Information
Dataset A (1)	Link for Dataset A (1) [1]	Creative Commons	559 images of class Neutral
Dataset A (2)	Link for Dataset A (2) [2]	Creative Commons	559 images of class Anger
Dataset B	Link for Dataset B [3][4][5]	Custom License "Terms of Use."	555 images of class Boredom and Engagement

Table 1 : Dataset Information

Data Cleaning

We used a number of strategies and procedures to preprocess the dataset during the "Data Cleaning" stage of our emotion recognition project to make sure it was consistent and of high quality for further analysis. The procedures followed, difficulties encountered, and illustrations of the cleaning effects before and after are included in this section.

Techniques and Methods

1. **Standardization of the Dataset**

   1.1. Resizing Images

   Resizing the photos to a consistent resolution was one of the first steps in data cleaning. This guaranteed data homogeneity while simultaneously lowering the computational cost of our model. Since **224 × 224** pixels is a standard input size for many deep learning models, we scaled every image to that size.

   1.2. Single format files

   We implemented file format conversion techniques to help make the model more resilient to changes in image format **png** to **jpg**. By doing so, we aimed to reduce the impact of file format change variations on the model's performance.

2. **Challenges and Solutions**

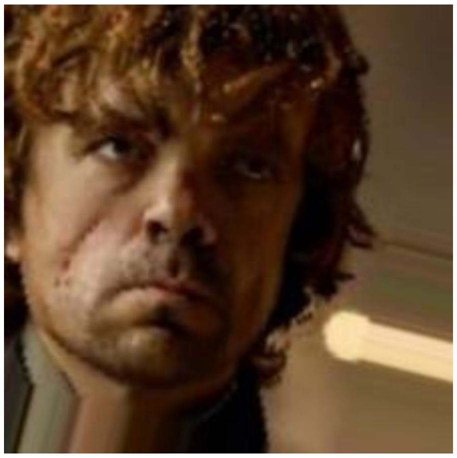
   2.1. Data Imbalance

   A problem with data imbalance arose throughout the data cleansing process. Our dataset's emotion categories were not evenly distributed, with considerably less samples for **boredom** and **engagement** emotions than for others. We created a more balanced dataset by under sampling the overrepresented categories and oversampling the underrepresented ones in order to remedy this problem.

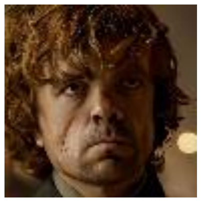
   2.2. Noise and Artifacts

   Our datasets contained images with noise, artifacts, and outliers that could adversely affect model training. To address this, we applied data augmentation techniques such as resize the image to reduce the impact of noise.

3. Example



Before



After

Figure 2. Image resize

Labeling

1. Merging Datasets and Class Mapping

1.1. Dataset Merging

Our project involved the combination of multiple datasets, each with its own set of emotion labels. Merging these datasets required mapping the emotion labels to a common set of emotional categories (Neutral, Engaged, Bored, Angry) to maintain consistency across the entire dataset. Dataset is stored in **CSV** file with fields file **name**, type of **class**, **size** of images, and file **format**.

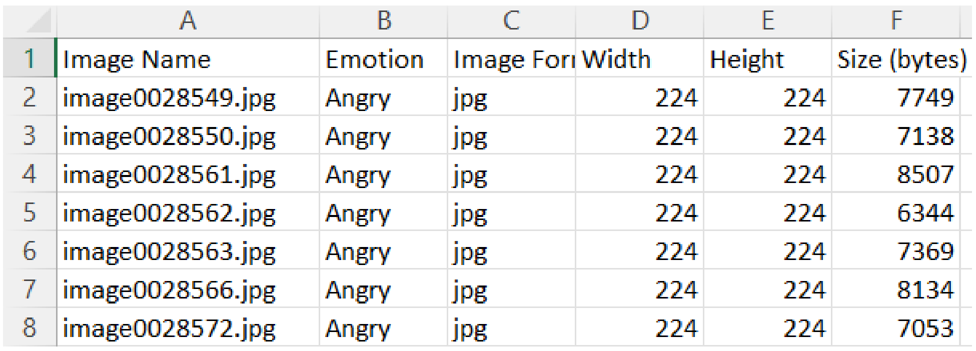
1.2. Challenges Faced

Inconsistent Labeling Schemes: Different source datasets had varying emotional labels. For example, one dataset used "Happiness" instead of "Engaged." This inconsistency posed a challenge in mapping labels accurately.

Data Skew: Some datasets had an uneven distribution of emotions, which required oversampling and under-sampling to achieve class balance.

1.3. Solution

All dataset is mapped to one single CSV file with related categories of class.



	A	B	C	D	E	F
1	Image Name	Emotion	Image Fori	Width	Height	Size (bytes)
2	image0028549.jpg	Angry	jpg	224	224	7749
3	image0028550.jpg	Angry	jpg	224	224	7138
4	image0028561.jpg	Angry	jpg	224	224	8507
5	image0028562.jpg	Angry	jpg	224	224	6344
6	image0028563.jpg	Angry	jpg	224	224	7369
7	image0028566.jpg	Angry	jpg	224	224	8134
8	image0028572.jpg	Angry	jpg	224	224	7053

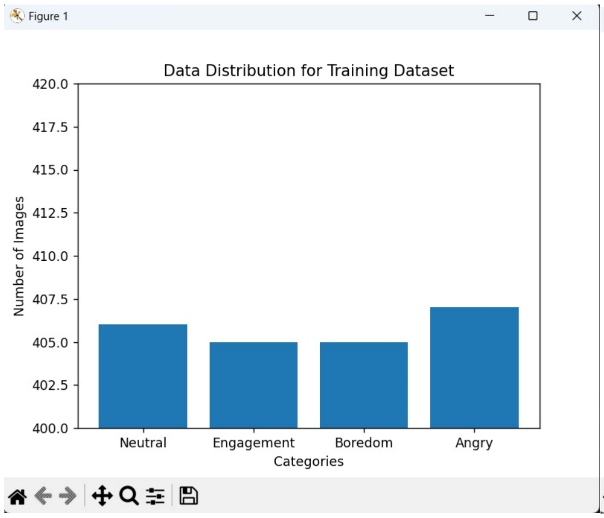
Figure 3. CSV file for image labeling

Dataset Visualization

In this section, we present visualizations that provide insights into our dataset. Effective visualization is crucial for understanding the data distribution, content, and potential challenges.

1. Class Distribution

To gain an understanding of the dataset's class distribution, we plotted a bar graph depicting the number of images in each emotion category. Class distribution is a critical factor to identify if any class is overrepresented or underrepresented. The following bar graph shows the class distribution in our dataset:



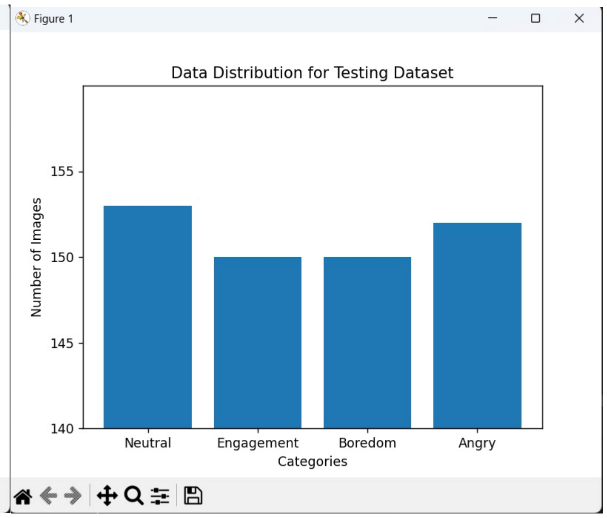


Figure 4, 5. Bar graph for Training and Testing images

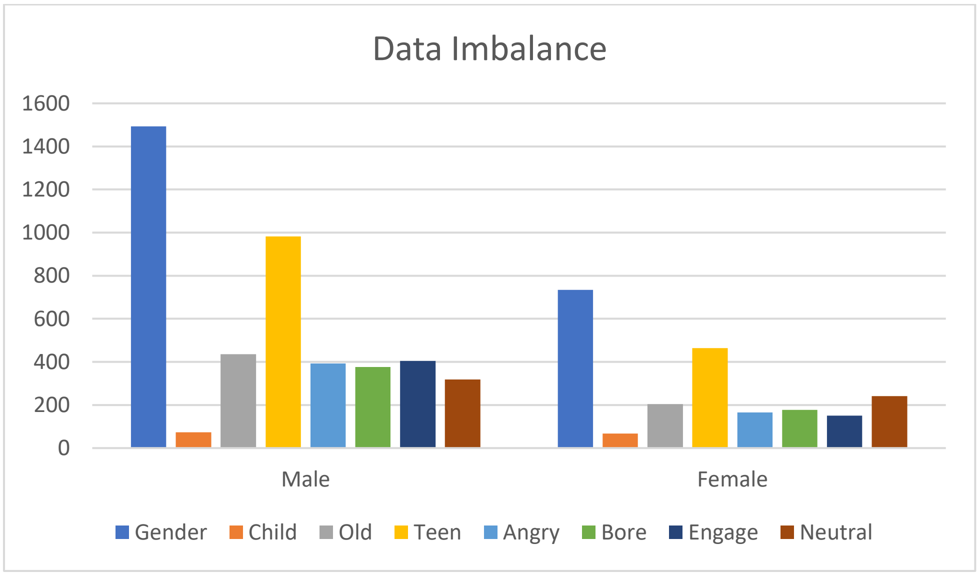


Figure 6. Bar graph for Final dataset

2. Sample Images

We randomly selected 25 images, random from each class, and arranged them in a 5x5 grid. We can use this visualization to gain insight into the range of facial expressions and check for any unusual patterns or incorrectly labeled data.

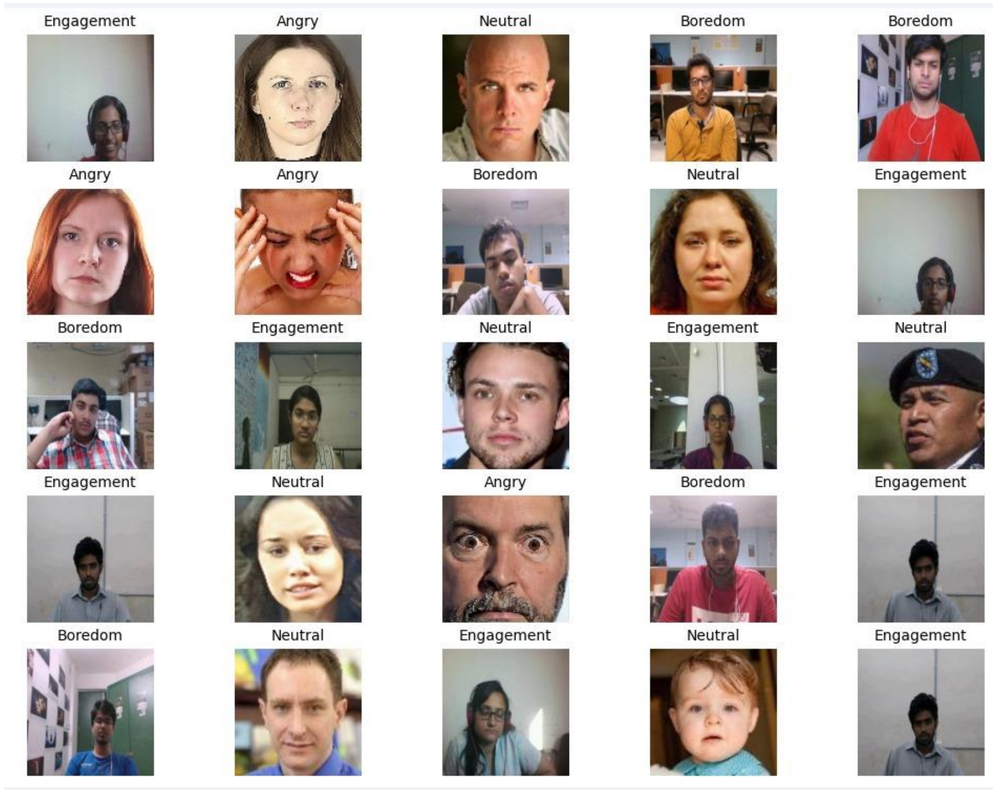


Figure 7. Labels for 25 random images

Analyzing these photographs, we can see different facial expressions and illumination, which are important components for identifying emotions. This also helps us validate the dataset's quality and maintain the accuracy of labeling.

3. Pixel Intensity Distribution

For color (RGB) images, we overlaid the intensity distributions of the Red, Green, and Blue channels on a single histogram for the same random 25 images.

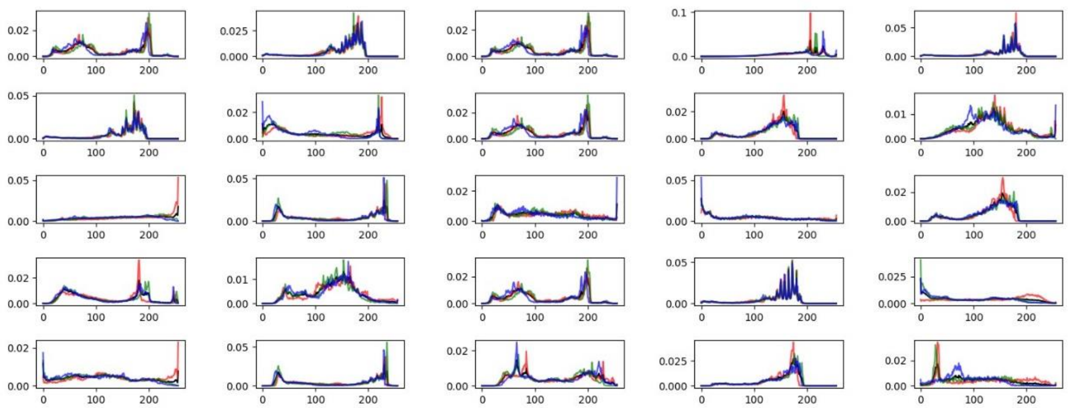


Figure 8. Pixel illumination for 25 random images

This histogram shows the distribution of pixel intensities for each color channel, offering insights into the range of lighting conditions within the dataset. Analyzing this information is essential for developing robust models capable of handling varying illumination levels.

CNN Architecture

In this phase, we delve into the technical aspects of the face recognition project called "A.I.ducation Analytics", specifically focusing on the implementation using PyTorch. Also, the dataset is used from phase 1. The following sections provide insights into the model architecture, training methodology, and evaluation metrics.

1. Model Overview and Architecture Details

The architecture is defined by the **ResNetModel** class. Convolutional layers (conversion_block), residual connections (self.res1 and self.res2), and a classifier make up this condensed form of a ResNet-like model.

The **forward()** method defines the flow of data through the layers.

Training Functions:

- fit_cycle() executes the training loop for a predetermined quantity of epochs.
- It includes training, validation, and evaluation steps, such as handling the optimizer, learning rate scheduler, and gradient clipping. evaluate() is used for model evaluation on validation and test sets.

Evaluation History:

- The code keeps track of training and validation accuracy and loss over epochs in history.

  Model Evaluation on Test Data:
- The trained model is evaluated on the test dataset.
- Confusion matrix, accuracy, and a classification report are generated to assess the model's performance.

Plotting Functions:

- plot_accuracies() and plot_losses() are helper functions to visualize accuracy and loss over epochs.

In the provided code, the CNN architecture consists of four convolutional layers:

First Convolution Block (self.conv1):

Input channels: 1 (grayscale images)
Output channels: 64
Kernel size: 3x3
Padding: 1
Batch normalization after convolution
ReLU activation

Second Convolution Block (self.conv2):

Input channels: 64 (output from previous layer)
Output channels: 128
Kernel size: 3x3
Padding: 1
Batch normalization after convolution
ReLU activation
Max pooling with a kernel size of 2x2 and stride 2

Residual Block 1 (self.res1):

Two consecutive convolutional layers:
Input channels: 128
Output channels: 128
Kernel size: 3x3
Padding: 1
Batch normalization after each convolution
ReLU activation
The output is added to the output of self.conv2

Third Convolution Block (self.conv3):

Input channels: 128 (output from the residual block)
Output channels: 256
Kernel size: 3x3
Padding: 1
Batch normalization after convolution
ReLU activation
Max pooling with a kernel size of 2x2 and stride 2

Fourth Convolution Block (self.conv4):

Input channels: 256 (output from the previous layer)
Output channels: 512
Kernel size: 3x3
Padding: 1
Batch normalization after convolution
ReLU activation
Max pooling with a kernel size of 2x2 and stride 2

Residual Block 2 (self.res2):

Two consecutive convolutional layers:
Input channels: 512
Output channels: 512
Kernel size: 3x3
Padding: 1
Batch normalization after each convolution
ReLU activation
The output is added to the output of self.conv4

Classifier (self.classifier):

Global Max pooling with a kernel size of 4x4
Flattening the output
Fully connected layer (Linear) with input features 512 (output channels from the last convolutional layer) and output classes 4.

2. Training Process

The training process in the provided code involves the following steps:

Data Loading and Preprocessing:

- The training and validation datasets are loaded using CDataset and split into batches using DataLoader.
- Images undergo preprocessing steps such as resizing, normalization, random cropping, and flipping through the specified transformations.

Model Definition and Initialization:

- The CNN model architecture (ResNetModel) is defined, specifying the number of input channels and output classes.
- The model is initialized with the defined architecture.

Device Handling and Optimization Configuration:

- The code checks for available GPUs and assigns the device accordingly (default_device()).
- The optimizer (Adam optimizer) is set up for training the model with specified parameters like weight decay.

Training Loop (fit_cycle()):

This loop runs for a specified number of epochs (epochs).
Within each epoch:

Training Phase:

- The model is set to training mode (model.train()).
- The training data loader (train_datal) is iterated batch-wise.
- **For each batch:**
  - ➢ **Forward pass:** Input batches are fed into the model (model.train_step(batch)) to obtain predictions.
  - ➢ **Loss calculation:** Cross-entropy loss is computed between predicted and actual labels.
  - ➢ **Backpropagation:** Gradients are calculated and backpropagated through the network.
  - ➢ **Optimization:** Optimizer updates the model parameters using the calculated gradients.
  - ➢ **Learning rate scheduling:** The Learning rate is adjusted based on the one-cycle learning rate scheduler.
  - ➢ **Metrics recording:** Training losses, learning rates, and accuracies are recorded for each batch.

Validation Phase:

- The model is set to evaluation mode (model.eval()).
- Validation data loader (val_datal) is iterated batch-wise.
- **For each batch:**
  - ➢ Validation step: Model predictions are obtained for validation data (model.validate_step(batch)).

    - ➢ Loss and accuracy calculation: Cross-entropy loss and accuracy are computed.
    - ➢ Metrics collection: Validation losses and accuracies are recorded.
- **Epoch End:**
    - ➢ After each epoch, the average training loss, validation loss, and validation accuracy are calculated and printed.
    - ➢ Training history (history) is updated with these metrics.

Evaluation on Test Data:

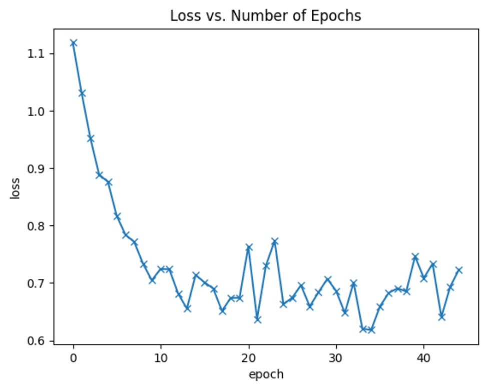
- After training is completed, the model is evaluated on the test dataset.
- Test data loader (test_data_loader) is used to iterate through test data.
- Predictions are made using the trained model, and evaluation metrics such as confusion matrix, accuracy, and classification report are generated.

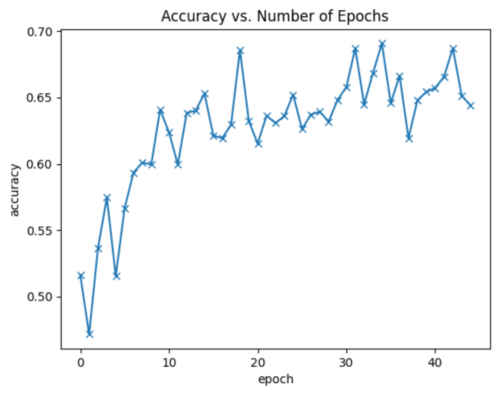
Visualization of Training Process:

- The code provides functions (plot_accuracies() and plot_losses()) to visualize the training and validation accuracies and losses over epochs using Matplotlib.

In summary, the training process involves iterating through the dataset in epochs, updating the model's parameters through backpropagation, adjusting learning rates, and monitoring performance metrics to assess the model's accuracy and loss.

3. Performance Analysis & Exploration





a) **b)**

Figure 9. a) Loss vs Number of epochs for Main Model: Kernal size = 3
b)Accuracy vs Number of epochs for Main Model: Kernal size = 3

Variant 1: Varying the Number of Convolutional Layers

We experiment by adding or removing convolutional layers in the architecture to observe how the depth of the network influences learning and generalization.

Changes Made:

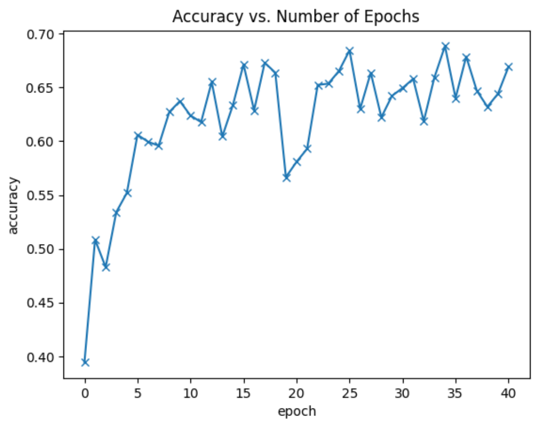
- Added an additional convolutional layer to the base architecture.

Reasoning:

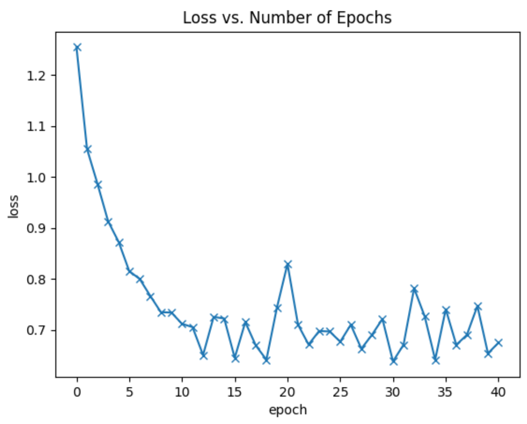
- Decrease depth may capture less complex hierarchical features.

Observations:

- Decrease training accuracy but potential underfit on the validation set.



a)



b)

Figure 10. a)Accuracy vs Number of epochs for Variant 1: Layer size = 3
b)Loss vs Number of epochs for Variant 1: Layer size = 3

Variant 2: Experimenting with Different Kernel Sizes:

We adjust the kernel sizes used in convolutional layers to understand the trade-offs regarding spatial granularity versus computational cost.

Changes Made:

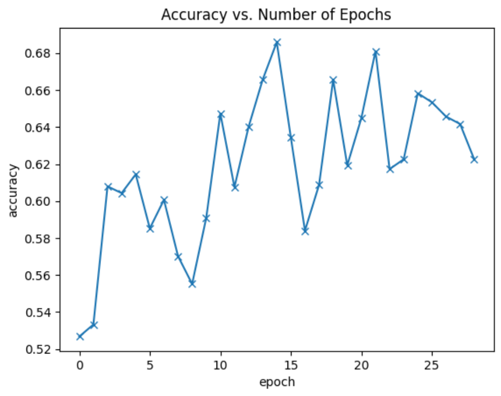
- Increased kernel size in one convolutional layer from 3 × 3 to 5 × 5.

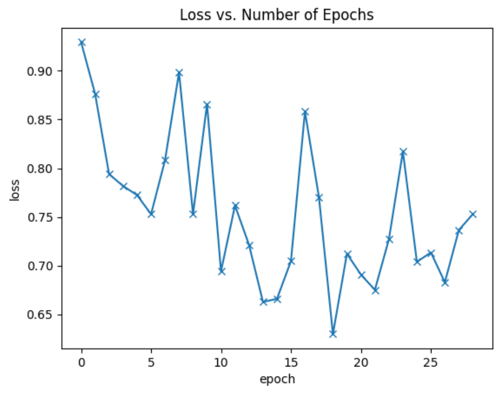
Reasoning:

- Larger kernel sizes capture broader features.

Observations:

- Improved recognition of broader features but increased computational cost.

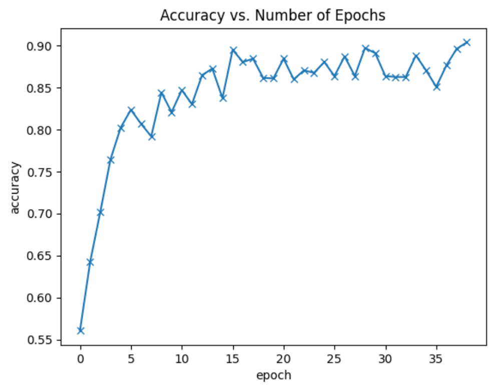


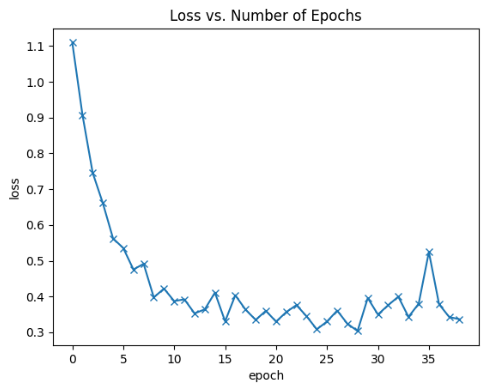


a) **b)**

Figure 11. a) Accuracy vs Number of epochs(12) for Variant 2: Kernal size = 5
b) Loss vs Number of epochs(12) for Variant 2: Kernal size = 5

Final Model: Experimenting with Different Images for boredom and engagement:





a) **b)**

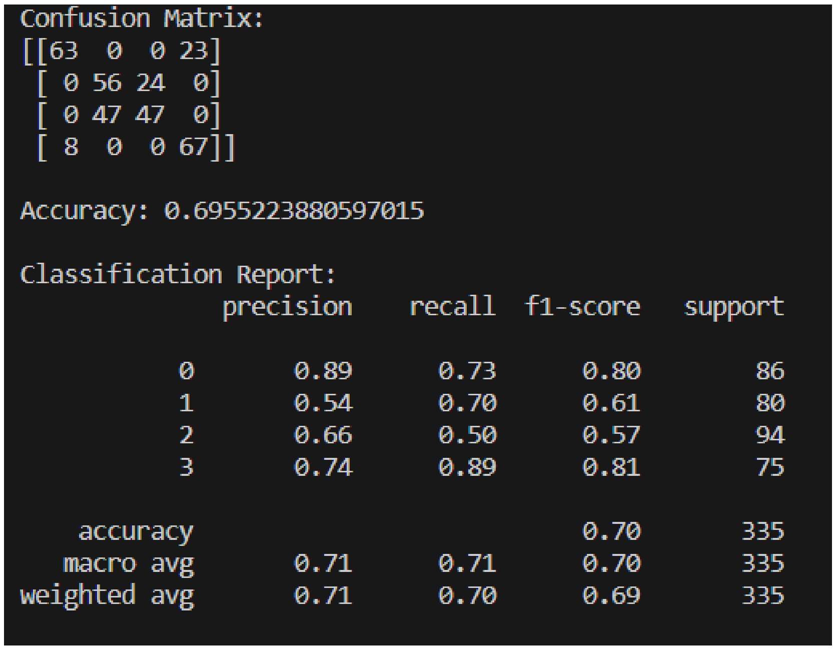
Figure 12. a) Accuracy vs Number of epochs for Final Model: Kernal size = 3
b) Loss vs Number of epochs for Final Model: Kernal size = 3

For this model train same provenance was used for image collection. Also, different variation selected compare to previous dataset like hand gesture, head movement etc.

Evaluation

All specific data is given in each category with respective output images from project. Using the same testing data, we assess the performance of the primary model and its two variations in this section. Confusion matrices are created as part of the evaluation process to visualize classification performance and to summarize important parameters including accuracy, precision, recall, and F1-score.

1. Performance Metrics, Confusion Matrix Analysis

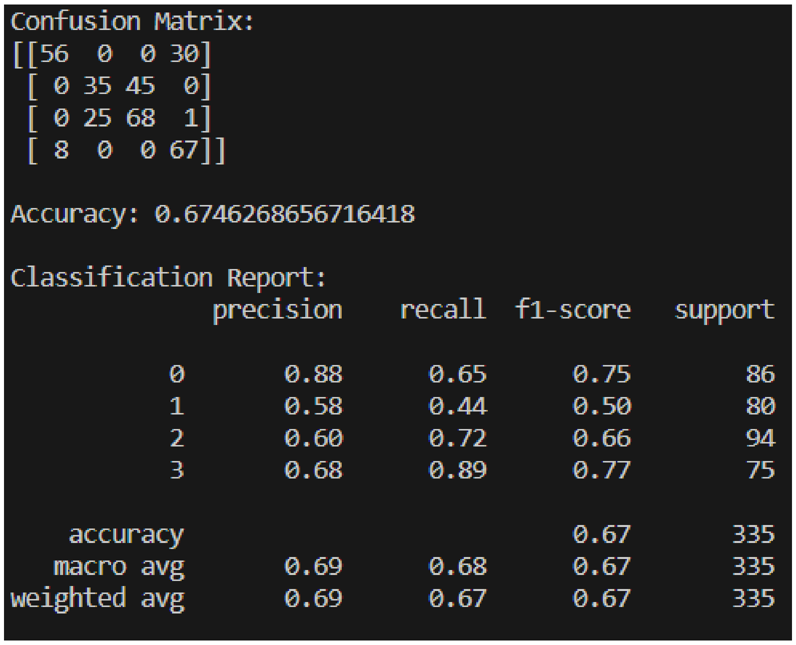


```
Confusion Matrix:
[[63  0  0 23]
 [ 0 56 24  0]
 [ 0 47 47  0]
 [ 8  0  0 67]]

Accuracy: 0.6955223880597015

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.73      0.80        86
           1       0.54      0.70      0.61        80
           2       0.66      0.50      0.57        94
           3       0.74      0.89      0.81        75

    accuracy                           0.70       335
   macro avg       0.71      0.71      0.70       335
weighted avg       0.71      0.70      0.69       335
```

Figure 13. Confusion, Performance Matrix for Main Model

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	125	1	0	26
Boredom	3	46	96	5
Engagement	0	90	60	0
Neutral	8	1	0	144

Table 2 : Confusion Matrix for Main Model

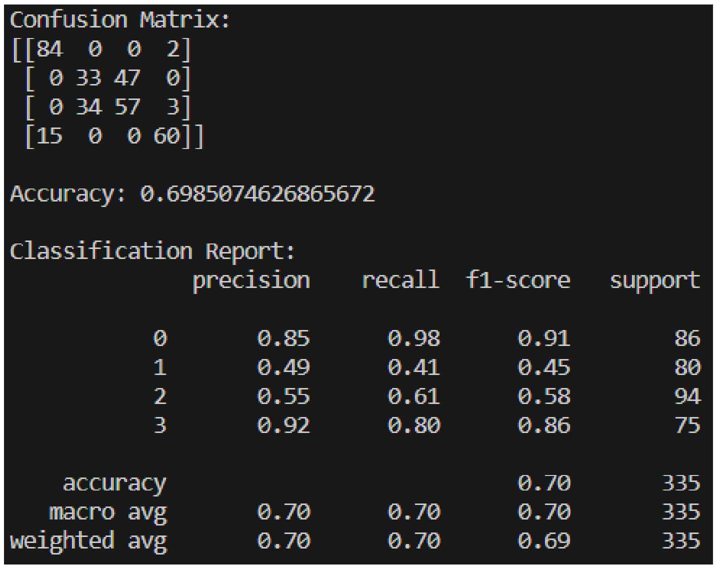


```
Confusion Matrix:
[[56  0  0 30]
 [ 0 35 45  0]
 [ 0 25 68  1]
 [ 8  0  0 67]]

Accuracy: 0.6746268656716418

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.65      0.75        86
           1       0.58      0.44      0.50        80
           2       0.60      0.72      0.66        94
           3       0.68      0.89      0.77        75

    accuracy                           0.67       335
   macro avg       0.69      0.68      0.67       335
weighted avg       0.69      0.67      0.67       335
```

Figure 14. Confusion, Performance Matrix for Variant 1

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	129	0	0	23
Boredom	1	47	102	0
Engagement	0	100	50	0
Neutral	7	0	0	146

Table 3 : Confusion Matrix for Varient 1

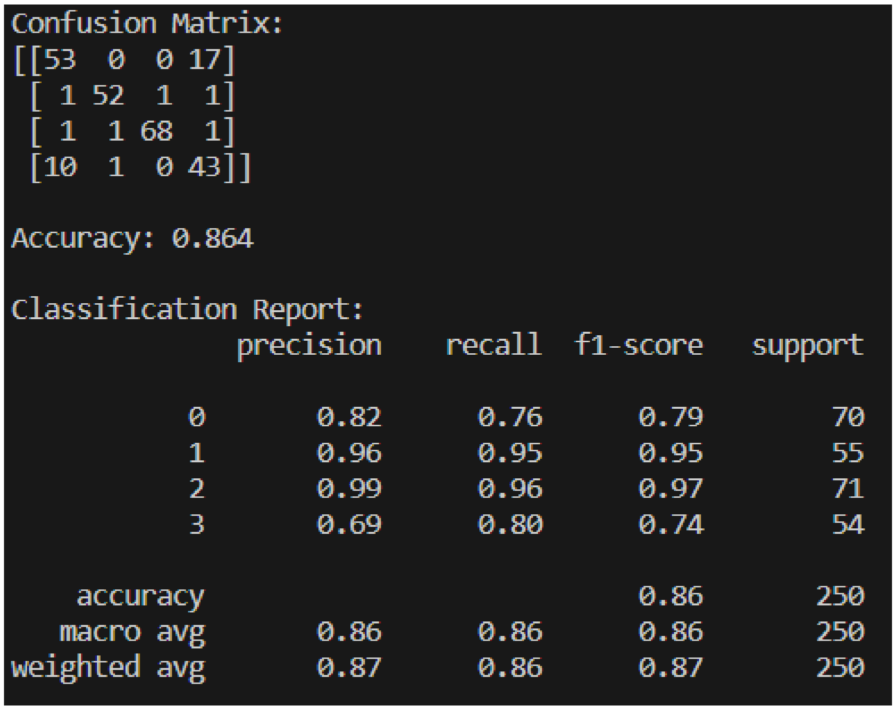


```
Confusion Matrix:
[[84  0  0  2]
 [ 0 33 47  0]
 [ 0 34 57  3]
 [15  0  0 60]]

Accuracy: 0.6985074626865672

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.98      0.91        86
           1       0.49      0.41      0.45        80
           2       0.55      0.61      0.58        94
           3       0.92      0.80      0.86        75

    accuracy                           0.70       335
   macro avg       0.70      0.70      0.70       335
weighted avg       0.70      0.70      0.69       335
```

Figure 15. Confusion, Performance Matrix for Variant 2

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	128	1	0	23
Boredom	0	42	99	7
Engagement	0	86	64	0
Neutral	5	0	0	148

Table 4 : Confusion Matrix for Varient 2



```
Confusion Matrix:
[[53  0  0 17]
 [ 1 52  1  1]
 [ 1  1 68  1]
 [10  1  0 43]]

Accuracy: 0.864

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.76      0.79        70
           1       0.96      0.95      0.95        55
           2       0.99      0.96      0.97        71
           3       0.69      0.80      0.74        54

    accuracy                           0.86       250
   macro avg       0.86      0.86      0.86       250
weighted avg       0.87      0.86      0.87       250
```

Figure 16. Confusion, Performance Matrix for Final Model

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	53	0	0	17
Boredom	1	52	1	1
Engagement	1	1	68	1
Neutral	10	1	0	43

Table 5 : Confusion Matrix for Final Model

By comparing all models, It is observed from the confusion matrix that the Main model is not good for differentiating between boredom and engagement images. So, based on those images, respective categories changed for better accuracy for the Final model. The Final dataset named Final_dataset and BiasTest_Dataset_P2 is for Phase 2 .

2. Impact of Architectural Variations

Model	Macro			Weighted			Accuracy
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
Main Model	0.61	0.62	0.61	0.62	0.62	0.62	0.62
Varient 1	0.61	0.61	0.61	0.62	0.61	0.61	0.61
Varient 2	0.62	0.63	0.62	0.63	0.63	0.63	0.63
Final Model	**0.86**	**0.86**	**0.86**	**0.87**	**0.86**	**0.87**	**0.864**

Table 6 : Performance Matrix for Main model

Number of Convolutional Layers:

- Convolutional Layer Count: Having more convolutional layers improves the network's comprehension of a wider variety of features and allows it to capture minute details in the data.
- Although deeper networks are better at capturing complexity, overfitting is more likely to occur.
- On the other hand, training with additional convolutional layers requires more time and computing power.
- The accuracy decreased little when the CNN layer was reduced.

Kernel Size:

- While a larger kernel concentrates on identifying more comprehensive patterns that are less important for this project, a smaller kernel excels at catching minute details, assisting in accurate class identification.
- Processing times are shortened and computational efficiency is increased with smaller kernels.
- The input gains scale variation invariance with larger kernels. Because smaller kernels perform well with little data, they reduce the risk of overfitting.
- Increasing Kernel size to 5 increased the accuracy from 61.98% to 63.14%

3. Conclusions and Forward look

When the kernel size was altered in comparison to the main model, a higher degree of accuracy was demonstrated. Variant 1 performed comparably better in all areas, including precision, accuracy, recall, f-measure, and computing efficiency.

Forward look

- Using a more extensive dataset improves model performance.
- Implementing precise classification for training sharpens the model's learning.

4. K-fold Cross-Validation

In this section, Main model and Final model evaluated for K-fold cross validation. Here, we used Stratified K-Fold. **StratifiedKFold** is a cross-validation technique used in machine learning to ensure that each fold or partition of the dataset is representative of the overall class distribution. It is particularly useful when dealing with imbalanced datasets where certain classes may be underrepresented. This method helps prevent issues where one or more classes have too few instances in either the training or testing set, leading to biased model evaluation[6].

Fold	Macro			Weighted			Accuracy
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
1	0.69	0.68	0.68	0.69	0.68	0.68	0.68
2	0.73	0.72	0.72	0.73	0.72	0.72	0.72
3	0.72	0.72	0.72	0.72	0.72	0.72	0.72
4	0.69	0.68	0.67	0.69	0.68	0.67	0.68
5	0.65	0.64	0.63	0.65	0.64	0.63	0.64
6	0.69	0.69	0.69	0.69	0.69	0.69	0.69
7	0.70	0.70	0.70	0.70	0.70	0.70	0.70
8	0.63	0.63	0.63	0.63	0.63	0.63	0.63
9	0.68	0.67	0.67	0.68	0.67	0.67	0.67
10	0.72	0.72	0.71	0.72	0.72	0.71	0.72
Average	**0.68**	**0.665**	**0.662**	**0.67**	**0.675**	**0.67**	**0.6849**

Table 7: K-fold for Main_Model.pth from Phase2

Fold	Macro			Weighted			Accuracy
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	
1	0.85	0.85	0.85	0.85	0.85	0.85	0.85029
2	0.90	0.90	0.90	0.91	0.90	0.90	0.9036
3	0.90	0.89	0.89	0.90	0.89	0.89	0.8915
4	0.84	0.83	0.83	0.84	0.83	0.83	0.8313
5	0.86	0.86	0.86	0.87	0.86	0.86	0.8614
6	0.85	0.85	0.85	0.86	0.86	0.85	0.8554
7	0.89	0.88	0.89	0.89	0.89	0.89	0.8855
8	0.89	0.88	0.88	0.89	0.89	0.88	0.8855
9	0.90	0.90	0.90	0.90	0.90	0.90	0.9036
10	0.84	0.82	0.82	0.84	0.83	0.82	0.8253
Average	0.872	0.876	0.877	0.875	0.88	0.877	0.8693

Table 8: K-fold for Final_Model.pth from Phase3

Observations:

- The Final model exhibits consistent performance across different folds compare to the Main model, with an average Macro F1 of 0.877 and weighted F1 of 0.877.

Comparison with Part II Evaluation:

- In comparison to the Main model from Part II, the k-fold cross-validation results show poor compared to the Final model in all aspects as shown in table 7 and 8.
- This analysis helps in understanding how the model's varying performance across different segments of the data affects the overall evaluation.

The Final_Model.pth demonstrates improved performance compared to the Main_Model.pth, with an average accuracy 0.8693.

Contrast with Original Train/Test Split (Phase 2):

Significant Improvement:

- The Final_Model.pth outperforms the Main_Model.pth in every metric, indicating that the model has learned more robust and generalizable patterns across the dataset.

Precision, Recall, and F1-Score:

- Precision, recall, and F1-scores are consistently higher for the Final_Model.pth, demonstrating better discrimination between classes and overall model effectiveness.

Possible Reasons for Discrepancies:

- **Data Distribution:** K-fold cross-validation provides a more comprehensive assessment of the model across various data segments, reducing the impact of biased splits.
- **Model Complexity:** The Final_Model.pth may have a more sophisticated architecture or better hyperparameter tuning, leading to improved performance.
- **More Training Data:** The k-fold cross-validation allows the model to train on a larger portion of the dataset, potentially improving its ability to generalize.

Summary:

The Final_Model.pth from Phase 3 consistently outperforms the Main_Model.pth from Phase 2 across all evaluation metrics. The k-fold cross-validation results provide a more robust assessment of the model's performance, showing improved consistency and generalization. The observed differences highlight the importance of thorough model evaluation using cross-validation to obtain a more accurate representation of performance across diverse data subsets.

Bias Analysis

1. Introduction

For bias analysis, we considered two attributes(age, and gender). Child, Teen, Old and Male, and Female categories were used for respective attributes. Attributes were selected based on available data distribution.

2. Bias Detection Results

Attribute	Group	Accuracy	Precision	Recall	F1-Score
Age	Child	0.9361	0.95	0.92	0.93
	Teen	0.7982	0.80	0.80	0.80
	Old	0.7359	0.75	0.76	0.75
	Average	**0.8234**	**0.8333**	**0.8267**	**0.8267**
Gender	Male	0.7816	0.79	0.79	0.79
	Female	0.8081	0.79	0.79	0.79
	Average	**0.7948**	**0.79**	**0.79**	**0.79**
Overall System Average		**0.8091**	**0.8116**	**0.8083**	**0.8083**

Table 9. Bias Analysis table for Main model (Phase2)

Age Group Analysis:

Child vs. Teen vs. Old:

- There is a substantial difference in accuracy between the "Child" and "Old" age groups, with the "Child" group having significantly higher accuracy (0.9361) compared to the "Old" group (0.7359).
- The precision and recall metrics also show differences, indicating potential disparities in how well the model performs for different age groups.

Gender Group Analysis:

Male vs. Female:

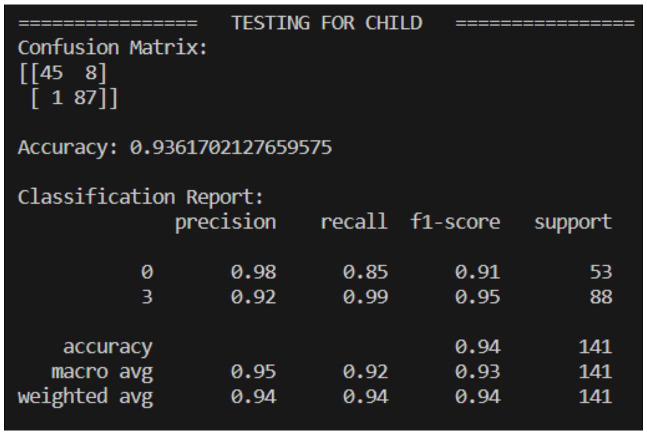
- The "Male" and "Female" gender groups have similar accuracy (0.7816 vs. 0.8081).
- The "Avg" row indicates a slight difference in average metrics between the two gender groups.

Overall System Analysis:

Average Metrics:

- The overall system average metrics (Accuracy, Precision, Recall, and F1-Score) show variations, suggesting that the model may exhibit disparities in performance across different attributes and groups.

3. Bias Mitigation Steps



```
================   TESTING FOR CHILD   ================
Confusion Matrix:
[[45  8]
 [ 1 87]]

Accuracy: 0.9361702127659575

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.85      0.91        53
           3       0.92      0.99      0.95        88

    accuracy                           0.94       141
   macro avg       0.95      0.92      0.93       141
weighted avg       0.94      0.94      0.94       141
```

Figure 17. Lack of boredom and Engagement data (Confusion Matrix)

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	45	0	0	8
Boredom	0	0	0	0
Engagement	0	0	0	0
Neutral	1	0	0	87

Table 10. Confusion Matrix for Child data for Main model (Phase2)

The main step taken by us is to add child data for lacking categories, boredom(20), and engagement(22). Also, based on previous phase2 confusion matrix results, we modified more variant data for most two miss classification categories like boredom and engagement maintaining image count.

Confusion Matrix	Angry	Boredom	Engagement	Neutral
Angry	22	0	0	6
Boredom	0	16	1	3
Engagement	3	0	15	4
Neutral	5	0	1	58

Table 11. Confusion Matrix for Child data for Final model (Phase3) (Accuracy : 0.8283)

After dataset change, we perform model train with same CNN architecture as Main model and created the Final_model with accuracy 0.8693.

4. Comparative Performance Analysis

Attribute	Group	Accuracy	Precision	Recall	F1-Score
Age	Child	0.8283	0.86	0.79	0.82
	Teen	0.9825	0.96	0.96	0.96
	Old	0.9277	0.94	0.94	0.94
	Average	**0.9128**	**0.92**	**0.8967**	**0.9067**
Gender	Male	0.9286	0.93	0.93	0.93
	Female	0.9420	0.92	0.92	0.92
	Average	**0.9353**	**0.925**	**0.925**	**0.925**
Overall System Average		**0.9240**	**0.9225**	**0.9108**	**0.9168**

Table 12. Bias Analysis table for Final model (Phase3)

By retraining the model based on the new dataset, the overall system average increased by 0.1 in each category.

Reference

[1]"Facial Expressions Training Data," www.kaggle.com. https://www.kaggle.com/datasets/noamsegal/affectnet-training-data?select=anger (accessed Oct. 27, 2023).
[2] "Facial Expressions Training Data," www.kaggle.com. https://www.kaggle.com/datasets/noamsegal/affectnet-training-data?select=neutral (accessed Oct. 27, 2023).
[3] "DAiSEE : Dataset for Affective States in E-Environments," people.iith.ac.in. https://people.iith.ac.in/vineethnb/resources/daisee/index.html
[4] "A Gupta, A DCunha, K Awasthi, V Balasubramanian, DAiSEE: Towards User Engagement Recognition in the Wild, arXiv preprint: arXiv:1609.01885"
[5] "A Kamath, A Biswas, V. Balasubramanian, A Crowdsourced Approach to Student Engagement Recognition in e-Learning Environments, IEEE Winter Conference on Applications of Computer Vision (WACV'16)"
[6] 3.1. Cross-validation: evaluating estimator performance. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/cross_validation.html#stratified-k-fold