COMP 472 ARTIFICIAL INTELLIGENCE

SmartClass A.I.ssistant

Team - OK_13
Chit Chit Myet Cheal Zaw (40110140)
Nadav Friedman (40091001)
Mehdi Chitsaz (40132819)

15 June 2024

Each team member's specialization:
Data Specialist: Chit Chit Myet Cheal Zaw
Training Specialist: Nadav Friedman
Evaluation Specialist: Mehdi Chitsaz

Github repository link - https://github.com/littleSquid00/COMP472_OB_13

We certify that this submission is the original work of group members and meets the Faculty's Expectations of Originality.

Chit Chit Myet Cheal Zaw	Nadav Friedman	Mehdi Chitsaz
(40110140)	(40091001)	(40132819)

DATASET

Overview of the existing dataset

For “Face expression recognition dataset”, there are “Train” and “Validation” folders.

The Train folder has 28,821 images and the Validation folder has 7,066 images. The thousands of images that these folders contain are pre-labeled. Each of the labels has its respective folder: angry, disgust, fear, happy, neutral, sad, and surprise. All the images are within their respective folders.

All of the images in this dataset are grayscale, frontal face shots, and 48x48px images.

For the “DAiSEE dataset”, there are “Test”, “Train” and “Validation” folders.

The Test, Train, and Validation folders contain 21, 70, and 22 subfolders respectively. Each of these subfolders represents a unique person, and the subfolder's name is thus a unique ID. Each subfolder identified by this unique ID contains hundreds of folders each representing a clip. Each clip folder contains a single video clip. The video clips are high resolution, coloured, and face the front of the person being filmed. In the background of the video, there are various objects, and sometimes other people. More information regarding the folder structure is present in the DAiSEE dataset's README.md file.

Dataset choices

We have four classes, Neutral, Angry, Engaged and Happy. We used 505 images for each class and that gives a total of 2020 images. In these 2020 images, most of them are frontal face shots.

We took Neutral, Angry and Happy images from “Face expression recognition dataset” found via the Kaggle website. We took Engaged images from the “DAiSEE” dataset.

The reason why I chose “Face expression recognition dataset” is that it is already pre-labelled Neutral, Angry and Happy, but the issue is that it has a lot of bad images with text on them and the resolution is not great. However, we think it is probably still a good choice because it prevents us from having to manually label thousands of images.

We chose the “DAiSEE” dataset because of its Engaged students' images. It is also already pre-labelled for us. The challenge was that it was videos of a different resolution and also they were not grayscale To overcome this, we had to write code to extract frames from the videos as well as to change the colours and the size.

Provenance Information

“Face expression recognition dataset”
https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset

“DAiSEE” dataset

“DAiSEE is a multi-label video classification dataset comprising of 9,068 video snippets captured from 112 users for recognizing the user affective states of boredom, confusion, engagement, and frustration "in the wild". The dataset has four levels of labels namely - very low, low, high, and very high for each of the affective states, which are crowd annotated and correlated with a gold standard annotation created using a team of expert psychologists.”
https://drive.google.com/file/d/1yrk_wyhZ-c7q0Mcyyi888ylFkl_JDELi/view

DATA CLEANING

The techniques and methods

(1) First, we extracted Engaged images from videos. We got first and last frames from the video by running the command python ExtractFrames.py

(2) Second, we cropped the images using Preview

(3) Third, we turned them into grayscale by running the following on Linux (ImageMagick is required) mogrify -colorspace Gray source.jpg

(4) Then, we resized the images to 48 x 48px for example by running the following on Linux (ImageMagick is required) mogrify -resize 48x48! source.jpg

(5) Last, we went through the 2000+ images manually to remove those with text on them, those with sunglasses, those that seem to have been mislabeled, and those that aren't real people (e.g: cartoons)

Note: for the facial recognition dataset, only step#5 is needed.

Challenges

For DAiSEE dataset, the script for extracting images from the video was not working (e.g.: `.DS_Store` files were provided - and that broke the script). Additionally, the source/destination directories were not easily configurable, so we had to rewrite parts of it.
Going through 2000+ images manually and removing those with text on them were also challenging because it took so much time.

Before-and-After Examples

(Initial picture)

(After turning into grayscale)

(After cropping)

(After resizing it into 48x48px)

LABELING

We chose the datasets that have images which are already labelled.

226 engaged pictures were taken from DAISEE dataset and the remainder were taken from the facial recognition dataset. DAISEE pictures were already labelled as engaged. The facial recognition one is already labelled for happy, angry and neutral classes.

Because DAiSEE dataset only has 226 engaged pictures, we took neutral images from the “Face expression recognition dataset”. We went through each of them and took images that looked engaged. We labelled them manually.

DATA VISUALIZATION

Class Distribution

To better understand the distribution of the images across our classes of facial expressions (Neutral, Angry, Happy, Engaged), we created a bar graph showing the number of images in each class. This is done to show whether any class is over or underrepresented. In our case, we have an equal number of images for each class.

Figure: Class Distribution

The bar chart above shows the number of images in each of our four classes of facial emotions. Each class contains 508 images, meaning our data is balanced and should contribute to more equal training. Of these 508 images in each class, 3 are images of team members, and the other 505 are from the collected data.

Pixel Intensity Distribution

For each class, we plotted the aggregate pixel intensity distribution. All our images were converted to black and white (grayscale) so we plotted a single histogram for each class. These plots help us to visualize the overall brightness and contrast characteristics of the images.

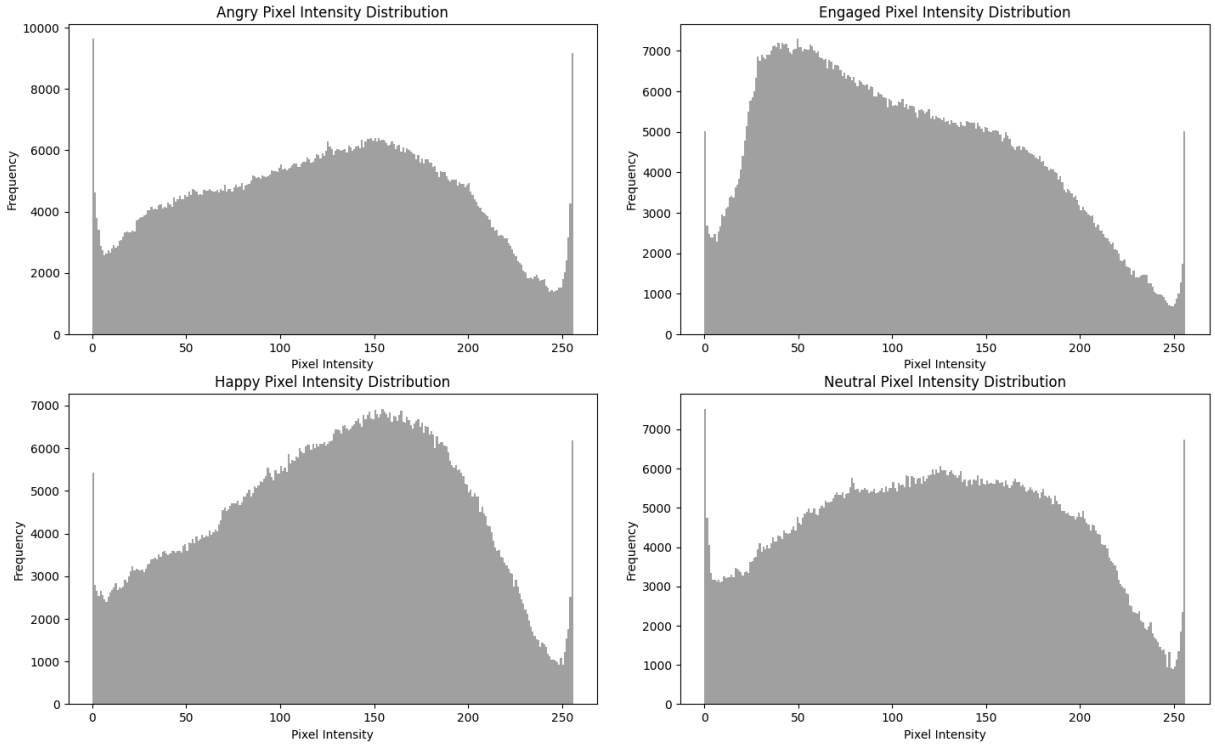


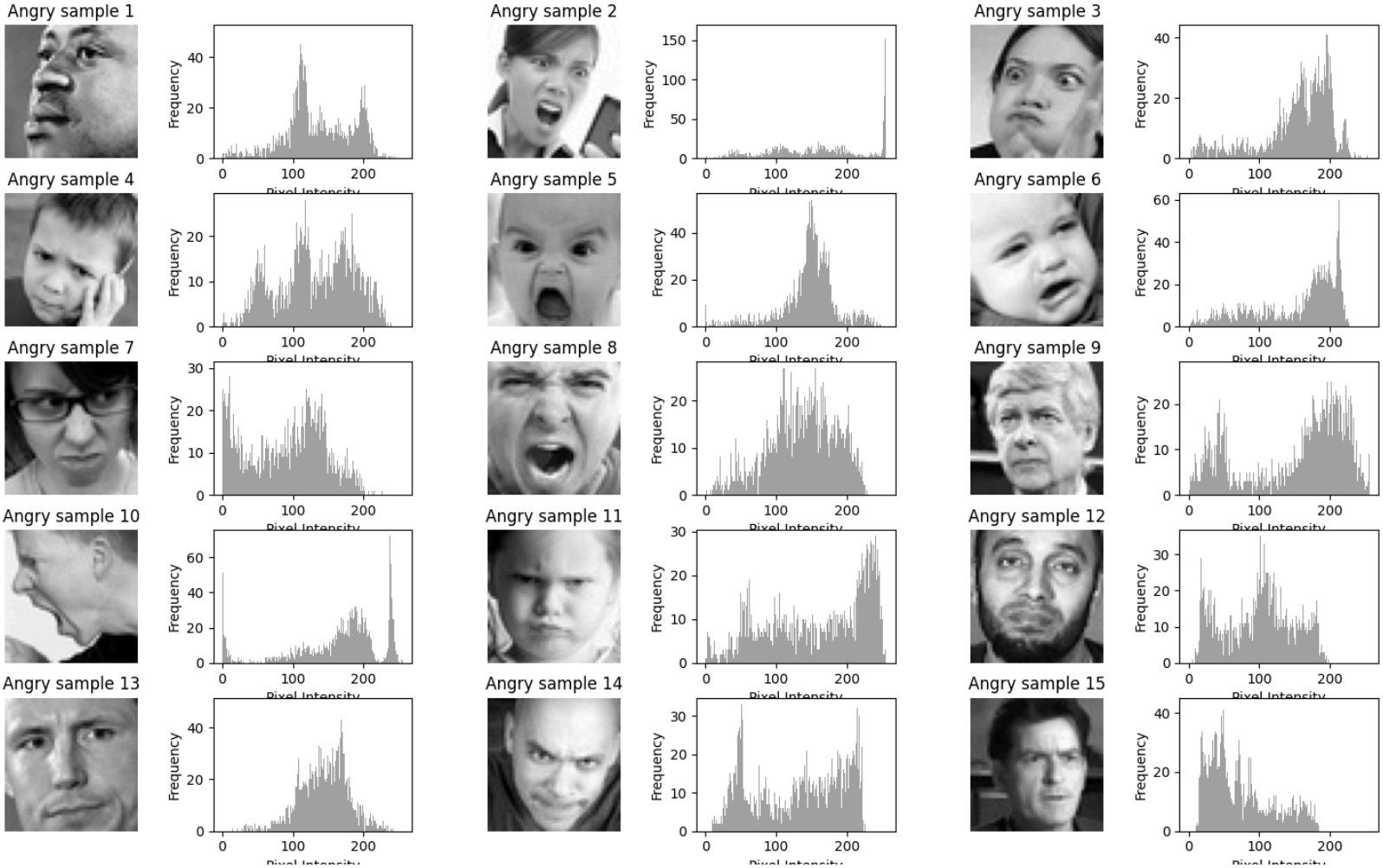
Figure: Pixel Intensity Distribution for each class

The histograms above display the pixel intensity distributions of the images in each class. They helped us understand the overall trend of brightness and contrast of the images. The fact that they are all in grayscale makes it easier to compare with different classes.

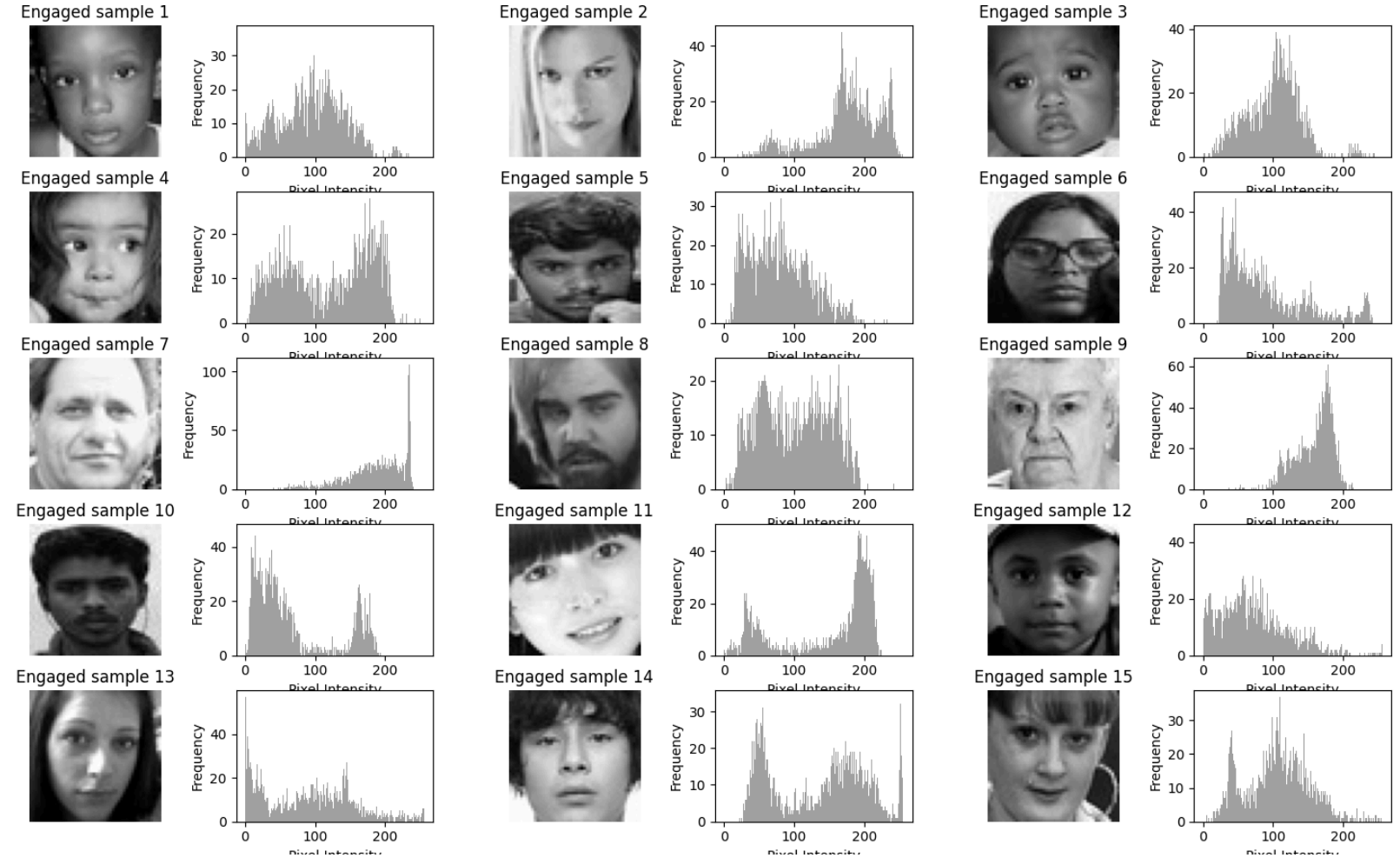
Sample Images

In this section, we present 15 randomly selected images from each class along with their pixel intensity histogram next to them laid in a 5 x 3 grid. The images are randomly chosen every time the DataVisualization.py script is run.

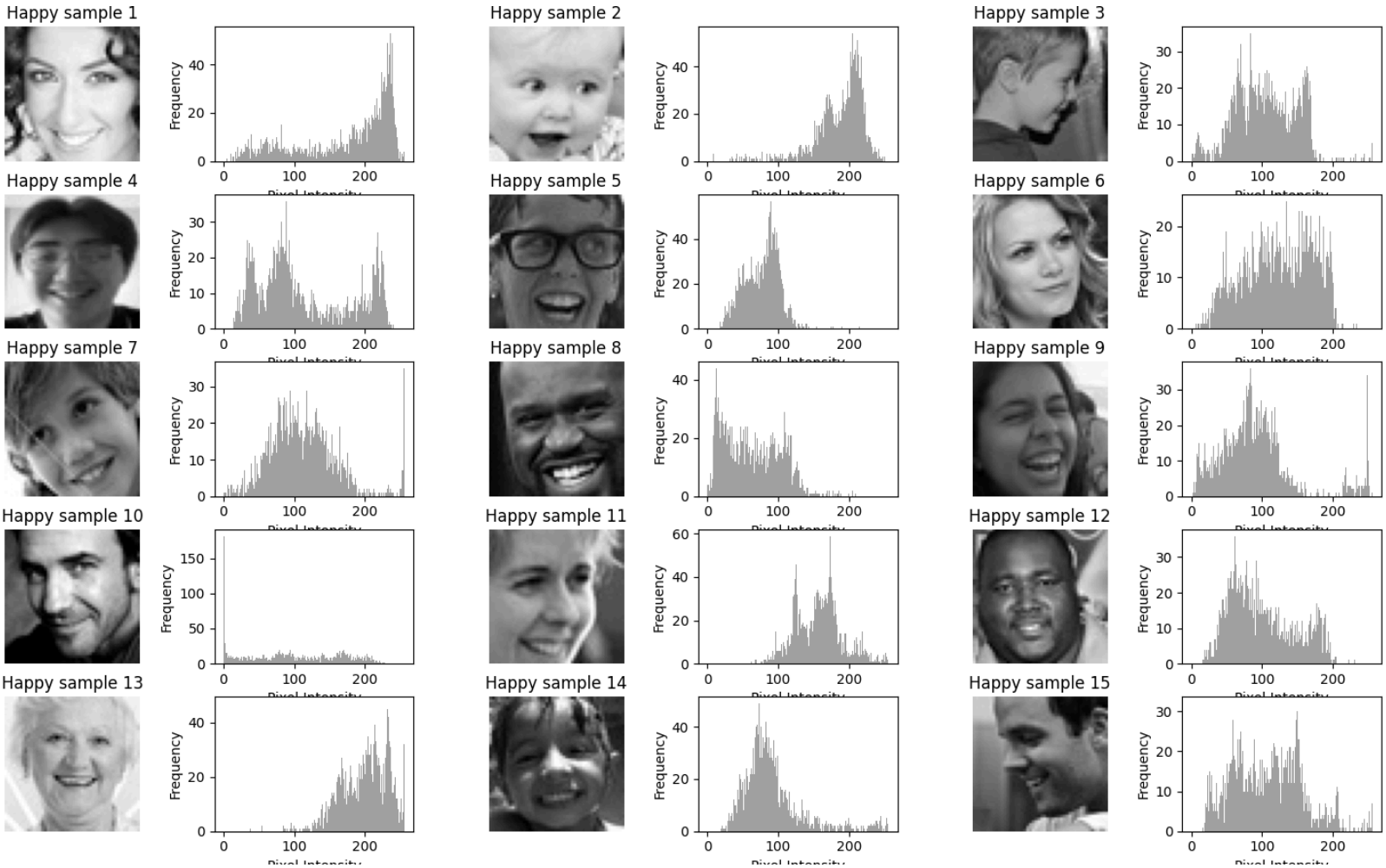
Angry


Angry sample 1
Angry sample 2
Angry sample 3
Angry sample 4
Angry sample 5
Angry sample 6
Angry sample 7
Angry sample 8
Angry sample 9
Angry sample 10
Angry sample 11
Angry sample 12
Angry sample 13
Angry sample 14
Angry sample 15
Frequency
Pixel Intensity

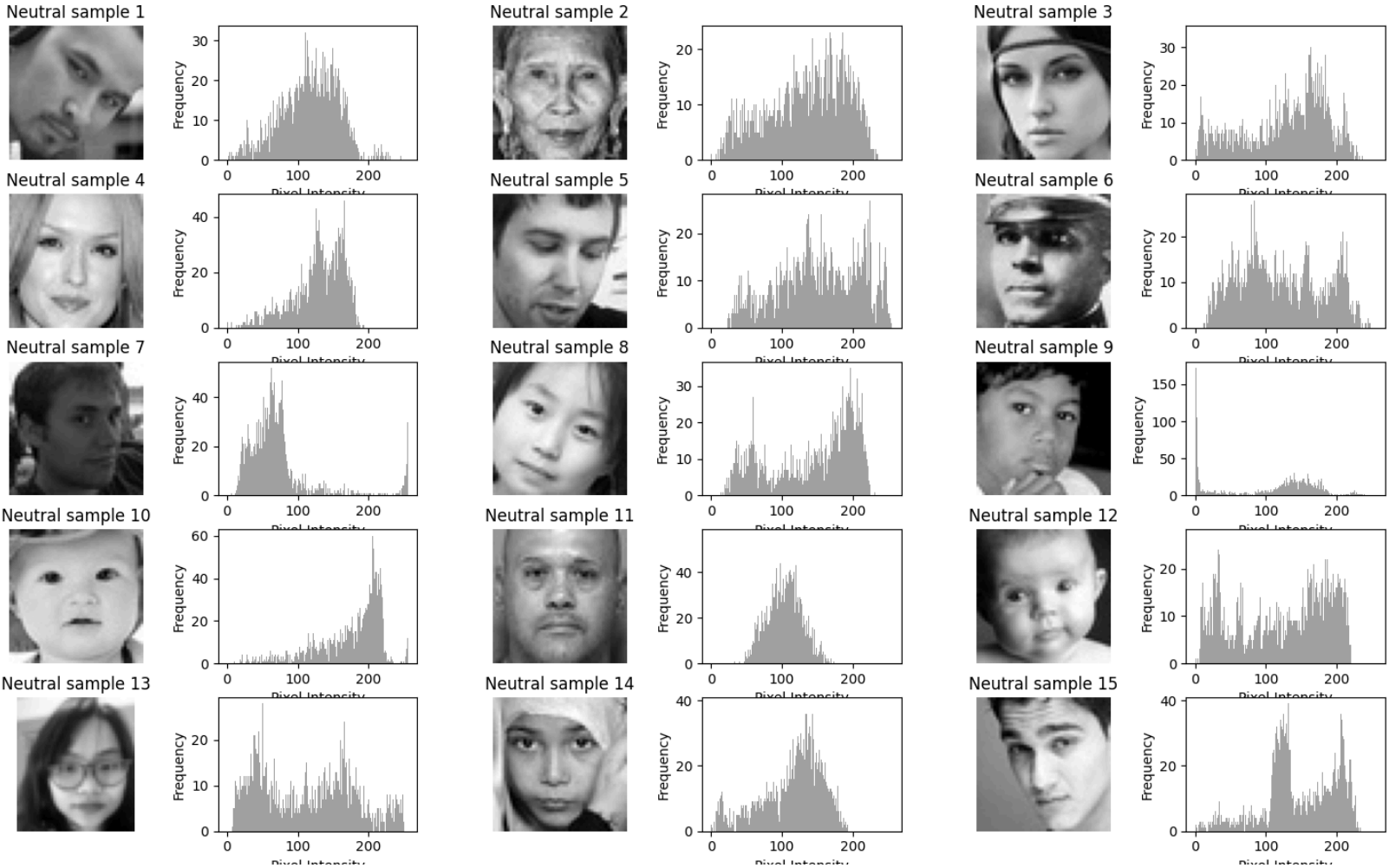
Engaged


Engaged sample 1
Engaged sample 2
Engaged sample 3
Engaged sample 4
Engaged sample 5
Engaged sample 6
Engaged sample 7
Engaged sample 8
Engaged sample 9
Engaged sample 10
Engaged sample 11
Engaged sample 12
Engaged sample 13
Engaged sample 14
Engaged sample 15
Frequency
Pixel Intensity

Happy


Happy sample 1
Happy sample 2
Happy sample 3
Happy sample 4
Happy sample 5
Happy sample 6
Happy sample 7
Happy sample 8
Happy sample 9
Happy sample 10
Happy sample 11
Happy sample 12
Happy sample 13
Happy sample 14
Happy sample 15
Frequency
Pixel Intensity

Neutral



The images and histograms in the grids above confirm that the preprocessing steps were done and that the images were correctly standardized and labele

CNN ARCHITECTURE

Model Overview and Architecture Details

The main model is a Convolutional Neural Network (CNN) designed for image classification tasks. It consists of convolutional layers followed by fully connected layers.

Input Layer:

Input size: 48x48 RGB images (3 channels).

Convolutional Layer 1:

Input channels: 3 (for RGB images)
Output channels: 32
Kernel size: 3x3
Padding: 1 (to preserve spatial dimensions)

Activation Function:
ReLU (Rectified Linear Unit)

Pooling Layer 1:
Type: Max pooling
Pooling size: 2x2 Stride: 2

Convolutional Layer 2:
Input channels: 32
Output channels: 64
Kernel size: 3x3
Padding: 1 (to preserve spatial dimensions)

Activation Function:
ReLU (Rectified Linear Unit)

Pooling layer2:
Pooling size: 2x2
Stride: 2

Fully connected layer1:
Units: 512

Activation Function:
ReLU (Rectified Linear Unit)

Dropout Layer:
0.5 (for regularization to prevent overfitting)

Fully connected layer2:
Units: 512

Regularization: Dropout layer with a dropout rate of 0.5 to reduce overfitting.
Activation Function: ReLU is used after each convolutional and fully connected layer to introduce non-linearity.
MaxPooling: Used after each ReLU activation to reduce the spatial dimensions and computational complexity.

Variant 1
Variant 1 of the CNN model introduces larger kernel sizes in the convolutional layers.

Convolutional Layer 1:
Filters: 32
Kernel size: 5x5
Padding: 2
Activation function: ReLU (Rectified Linear Unit)

Convolutional Layer 2:
Filters: 64
Kernel size: 5x5
Padding: 2
Activation function: ReLU (Rectified Linear Unit)

Other layers and parameters remain the same as the main model.

Variant 2
Variant 2 adds an additional convolutional layer with smaller kernel sizes.

Convolutional Layer 1:
Filters: 32
Kernel size: 3x3
Padding: 1
Activation function: ReLU (Rectified Linear Unit)

Convolutional Layer 2:
Filters: 64
Kernel size: 3x3
Padding: 1
Activation function: ReLU (Rectified Linear Unit)

Convolutional Layer 3:
Filters: 128
Kernel size: 3x3
Padding: 1
Activation function: ReLU (Rectified Linear Unit)

Pooling layer3:

Type: Max pooling
Pooling size: 2x2
Stride: 2

Fully connected layer1:
Unit: 512
Activation function: ReLU (Rectified Linear Unit)
Dropout: 0.5

Output layer:
Units: Number of classes (4)
Activation function: Softmax

Training Process

The following details outline the training process and methodology used for training three different CNN models: the main model (CNN), CNNVariant1, and CNNVariant2.

Training Methodology:

Batch size: 64

Learning rate: 0.001

Number of epochs: 20

Patience: Early stopping is triggered if the validation loss does not improve for 5 epochs, preventing overfitting and unnecessary computation.

Transform: The input images are subjected to data augmentation using random horizontal flips. This helps in making the model more robust by providing varied representations of the input data.

Normalization: The images are normalized with mean and standard deviation of (0.5, 0.5, 0.5) for each channel, which helps in faster convergence.

Deataset: The data set is loaded from a directory structure using ./processed_data/alldataset/xxx.jpg

Splitting: The dataset is split into training (70%), validation (15%), and test sets (15%) using indices.

Optimization Algorithms and Techniques

Adam optimizer: The Adam optimizer `optim.Adam` is used for updating the model parameters. Adam is chosen due to its efficiency and effectiveness in handling sparse gradients and adaptive learning rates.

Mini-batch gradient descent: Training is performed using mini-batches, which allows for efficient use of GPU memory and leads to faster convergence compared to using the entire dataset for each update.

Early stopping: Early stopping is implemented to prevent overfitting. If the validation loss does not improve for 5 consecutive epochs, training is stopped early.

Model Saving: The model with the lowest validation loss is saved during training. This ensures that the best-performing model on the validation set is retained.

EVALUATION

Performance Matrix

Model	Macro			Micro			Accuracy
	P	R	F	P	R	F	
Main Model	0.6372	0.6305	0.6285	0.6305	0.6305	0.6305	0.6305
Variant 1	0.6623	0.6576	0.6572	0.6576	0.6576	0.6576	0.6576
Variant 2	0.6854	0.6712	0.6742	0.6712	0.6712	0.6712	0.6712

Accuracy: Variant 2 has the highest accuracy (0.6712), followed by Variant 1 (0.6576), and then the Main Model (0.6305). Higher accuracy indicates that Variant 2 performs better overall in correctly classifying the emotions compared to the other models.

Precision: Variant 2 also leads in both macro and weighted precision (0.6854). Variant 1 follows with a macro and weighted precision of 0.6623. The Main Model has the lowest precision (0.6372).Higher precision indicates that when the model predicts a certain emotion, it is more likely to be correct, reducing false positives. This is particularly useful in applications where false alarms need to be minimized.

Recall: Variant 2 again leads in recall (0.6712), suggesting it is better at identifying all relevant instances of emotions, reducing false negatives. Variant 1 has a recall of 0.6576, and the Main Model has the lowest recall (0.6305). High recall is crucial in contexts where missing an emotional cue could be detrimental, such as in mental health monitoring where every sign of distress should be detected.

F1-score: balance between precision and recall. Variant 1 follows with an F1-score of 0.6572 (macro and weighted). The Main Model has the lowest F1-score (0.6285). The F1-score is a balanced measure, and higher values suggest a model that performs well overall in both precision and recall.

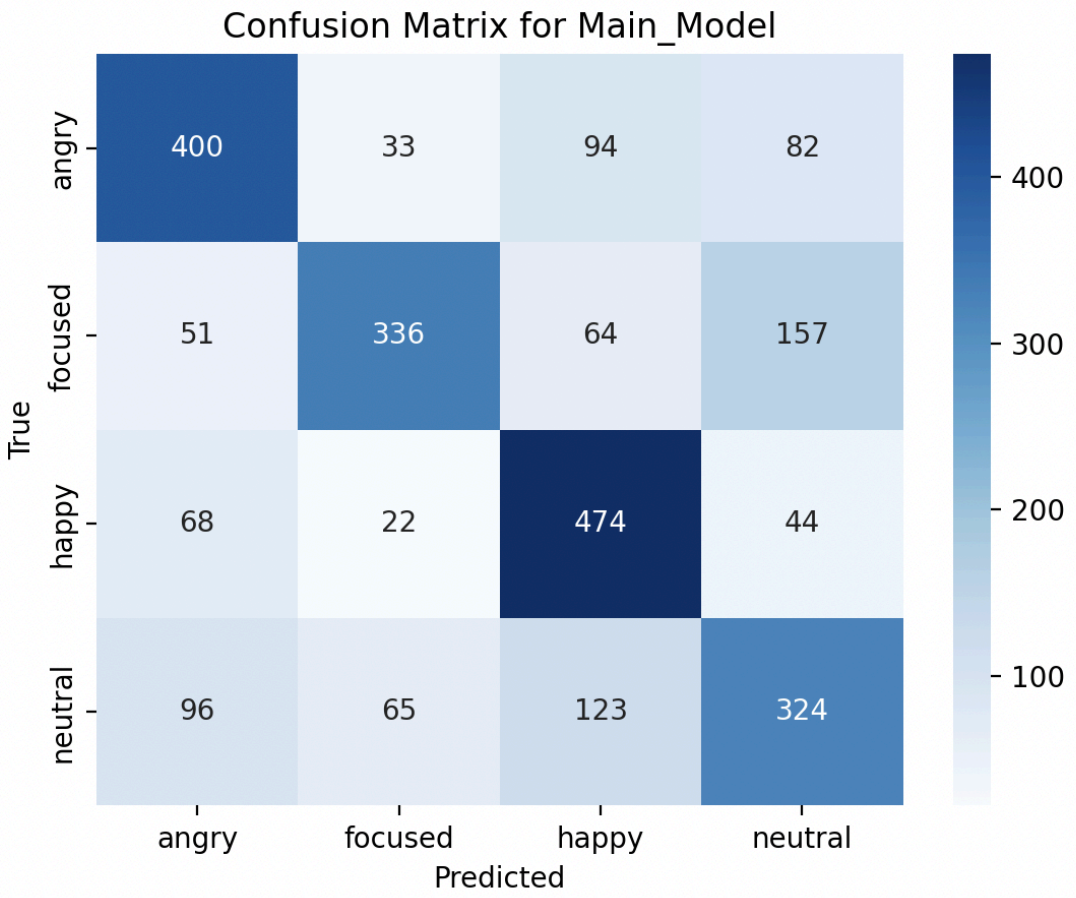
Classification Report for Main_Model:

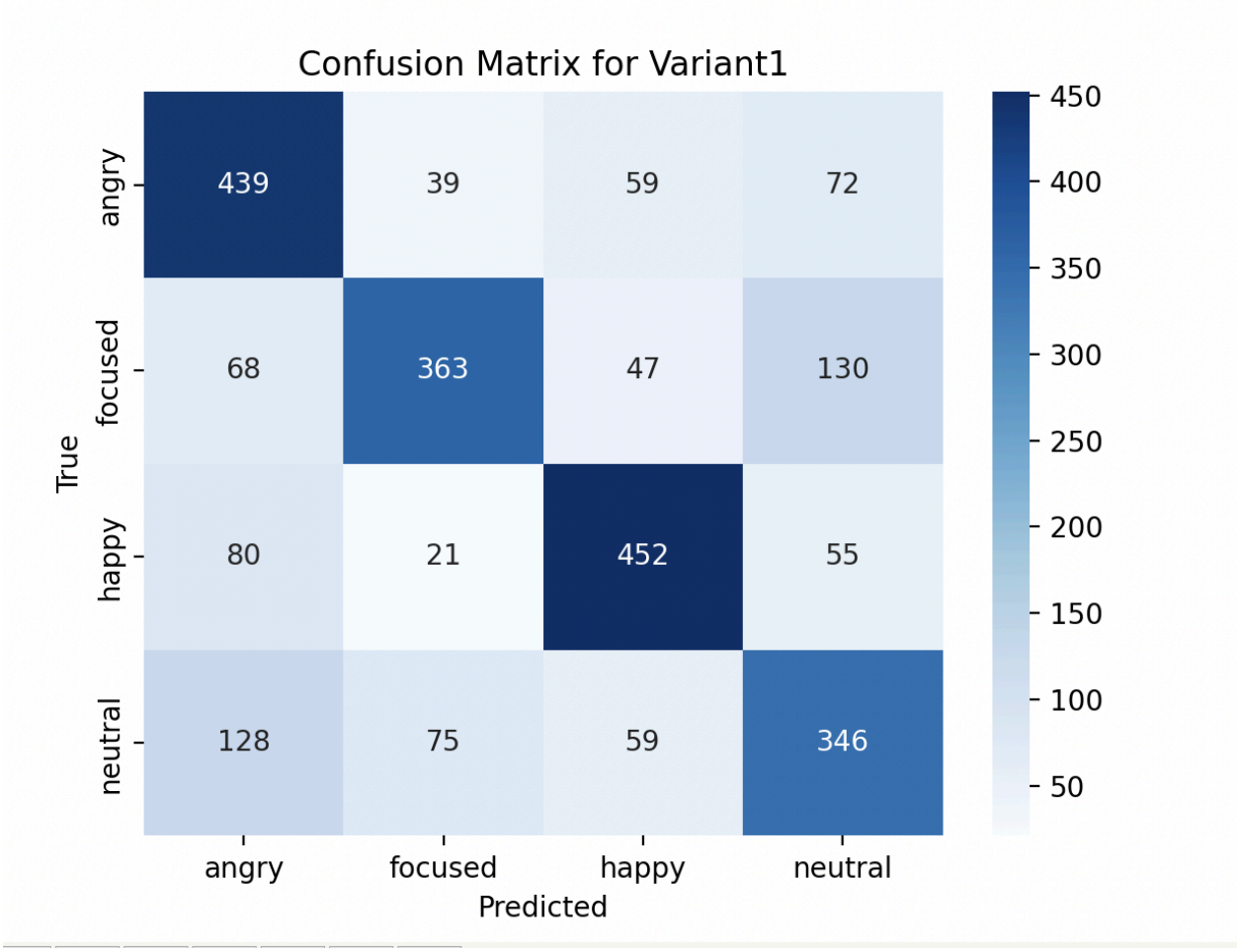
	precision	recall	f1-score	support
angry	0.6504	0.6568	0.6536	609
focused	0.7368	0.5526	0.6316	608
happy	0.6278	0.7796	0.6955	608
neutral	0.5338	0.5329	0.5333	608
accuracy			0.6305	2433
macro avg	0.6372	0.6305	0.6285	2433
weighted avg	0.6372	0.6305	0.6285	2433

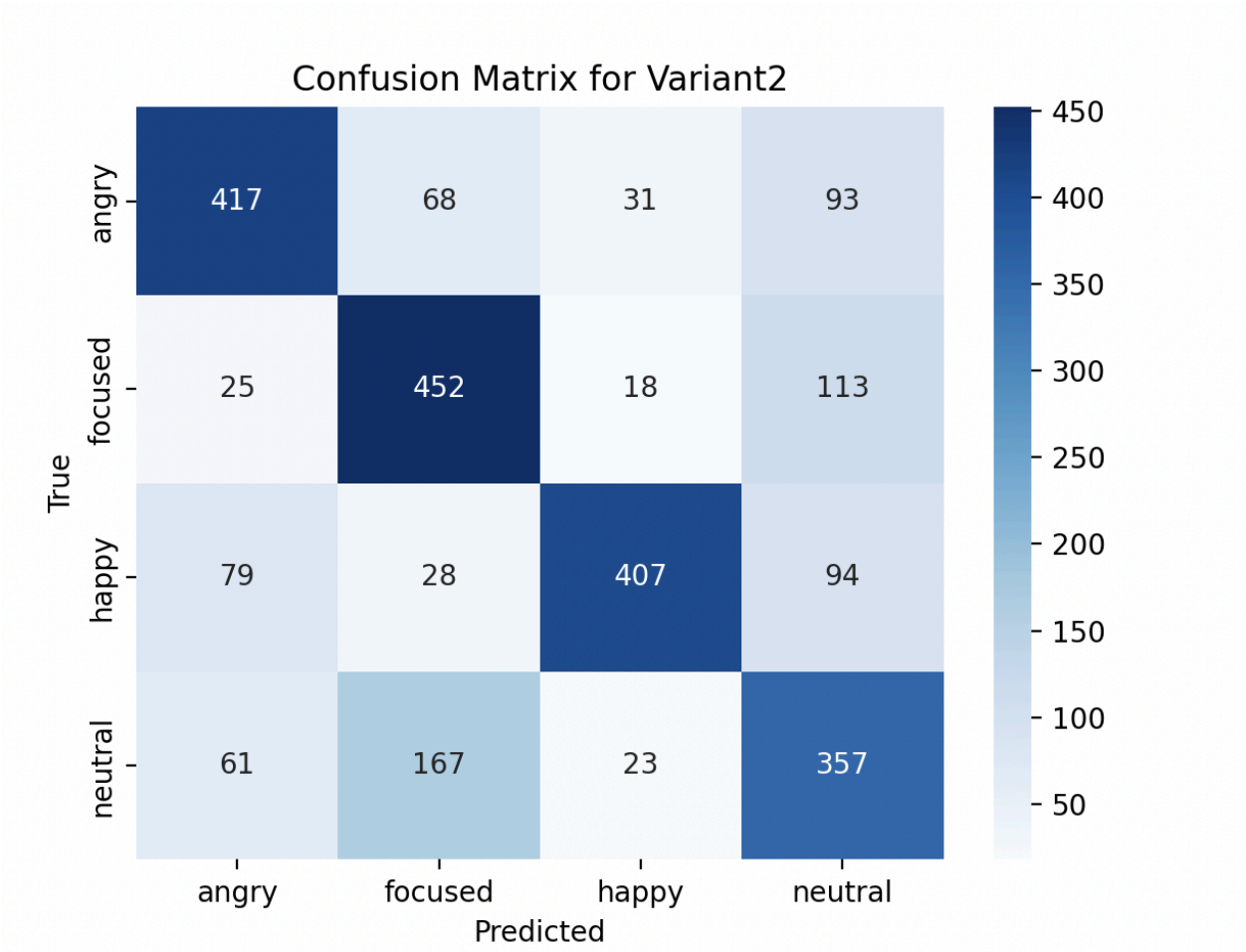
```
Classification Report for Variant1:
              precision    recall  f1-score   support

       angry     0.6140    0.7209    0.6631       609
     focused     0.7289    0.5970    0.6564       608
       happy     0.7326    0.7434    0.7380       608
     neutral     0.5738    0.5691    0.5714       608

    accuracy                         0.6576      2433
   macro avg     0.6623    0.6576    0.6572      2433
weighted avg     0.6623    0.6576    0.6572      2433
```

```
 Classification Report for Variant2:
              precision    recall  f1-score   support

       angry     0.7165    0.6847    0.7003       609
     focused     0.6322    0.7434    0.6833       608
       happy     0.8497    0.6694    0.7489       608
     neutral     0.5434    0.5872    0.5644       608

    accuracy                         0.6712      2433
   macro avg     0.6854    0.6712    0.6742      2433
weighted avg     0.6854    0.6712    0.6742      2433
```

Confusion Matrix Analysis


Confusion Matrix for Main_Model
angry
focused
happy
neutral
True
400
33
94
82
51
336
64
157
68
22
474
44
96
65
123
324
angry
focused
happy
neutral
Predicted
400
300
200
100


Confusion Matrix for Variant1
True
Predicted
angry
focused
happy
neutral
439
39
59
72
68
363
47
130
80
21
452
55
128
75
59
346
450
400
350
300
250
200
150
100
50



As you can see from the confusion matrix for the main model, Variant 1 and Variant 2, Class Neutral is often confused with Class Focused, it might be due to their visual similarities or due to the dataset.

Class Happy is consistently well-recognized, it could be because of distinctive and easy-to-learn features or better representation in the training data.

Impact of Architecture Variations

Small kernels: Smaller kernels are excellent at capturing fine details and local patterns. They are effective at detecting small, intricate features like edges, corners, and fine textures, which are crucial for recognizing detailed facial features.

Large kernels: Larger kernels capture broader patterns and more contextual information within a single layer. They are useful for recognizing larger structures and more spatially extended features within the image.

The architecture changes made for variation 2 model (i.e. smaller kernel size) improved performance because a smaller kernel like 3×3 is more effective in detecting edges compared to a larger kernel because it can capture the gradient changes more precisely.

Conclusions and Forward Look

Best model

Variant2 demonstrated the highest overall accuracy (0.6712) and macro average F1-score (0.6742). It showed notable performance across most classes, with particularly high precision and F1-score for the 'happy' class. Variant2 maintained a good balance between precision and recall, leading to higher F1-scores. This balance is crucial for ensuring that the model performs consistently well across all classes. Variant2 excelled in detecting 'focused' and 'happy' emotions, which might be due to better feature extraction or more effective handling of these specific classes in the training process.

Future refinements

Perform extensive hyperparameter optimization using techniques like grid search or Bayesian optimization to fine-tune model parameters for better performance. Use data augmentation to create synthetic samples, particularly for classes that are harder to detect, to help the model generalize better.

REFERENCES

1. J.Oheix. (2019), “The facial expression recognition dataset” Source.[Online]. Available: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset

2. A. Gupta et al.(n.d.), ”DAiSEE dataset” Source. [Online]. Available: https://drive.google.com/file/d/1yrk_wyhZ-c7q0Mcyyi888ylFkl_JDELi/view

3. J.Hunter et al. (n.d), “Matplotlib Documentation” Source. [Online]. Available: https://matplotlib.org/stable/contents.html

4. “OpenCV Documentation” Source. [Online]. Available: https://docs.opencv.org/master/