Gina Cody School of Engineering and Computer Science

COMP 6721 Applied Artificial Intelligence (Fall 2023)

Project Assignment, Part I

Submitted To: Professor **Dr. René Witte**

Submitted By:

Team Name	**NS_14**	
Student Name	**ID**	**Specialization**
Anurag Agarwal	40232644	Data Specialist
Vidhi Sagathiya	40232374	Training Specialist
Jimi Mehta	40225526	Evaluation Specialist

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

GitHub Repository: ***AIDucation Analytics***

Dataset

We have incorporated the "AffectNet Training Data" dataset, sourced from Kaggle, as part of our project. This dataset contains facial images with associated emotions. Here is an overview of this dataset:

Overview

AffectNet is a large database of faces labeled by "affects" (psychological term for facial expressions). To accommodate common memory constraints, the resolution was reduced to 96x96. Meaning that all images are exactly 96x96 pixels.

Total Images	**14071**
Class Name	**Number of Images**
Anger	3219
Engaged/ Focused	2780
Bored/ Tired	3092
Neutral	4980

Characteristics of Dataset

The "AffectNet Training Data" dataset is notable for its diverse collection of facial expressions and emotions. It encompasses a wide range of emotions, including happiness, sadness, anger, and neutrality, making it a comprehensive resource for emotion recognition tasks. The dataset features images with varying backgrounds, lighting conditions, and poses, providing a realistic representation of facial expressions in different real-world scenarios. It includes both frontal face shots and images with varying head orientations, enhancing its suitability for training AI models to recognize emotions in diverse contexts. This diversity in emotions and environmental factors makes the dataset an asset for building robust and versatile facial emotion recognition systems.

Justification for Dataset Choices

In the pursuit of developing an effective AI system for emotion recognition and object classification, the selection of appropriate datasets plays a pivotal role in the success of the project. The dataset chosen for this project has been carefully selected based on its relevance, diversity, and the challenges it may present. This section provides a detailed justification for the dataset choice.

The "AffectNet Training Data" dataset from Kaggle was a natural choice for our project due to its close alignment with our primary task of emotion recognition. This dataset offers a substantial collection of facial images, each labeled with a specific emotion [1].

The relevance of this dataset to our project is evident as it provides a comprehensive range of emotions, including happiness, sadness, anger, and neutrality, among others [1]. The substantial size of this dataset, with 15000 images, ensures that our AI model has access to a rich variety of emotional expressions, making it ideal for training a robust emotion recognition system [1].

Also, by using Singular Value Decomposition, each image's Principal Component Analysis was calculated. The threshold for the "percentage of the first component (index 0) in the principal components" (in short, the PFC%) was set to lower than 90%. This means that most if not all the monochromatic images were filtered out.

Challenges faced

Challenges posed by the "AffectNet Training Data" dataset primarily include data diversity, data management and class imbalance.

Data Diversity: The dataset's diversity in terms of lighting conditions, facial orientations, and backgrounds presented challenges. Variability in real-world conditions can affect the model's ability to generalize. Strategies to ensure the model's robustness across diverse scenarios were necessary.

Data Management: The dataset is substantial, containing many images. Managing and processing this volume of data efficiently was a significant challenge. Effective data preprocessing techniques were essential to ensure the dataset was well-prepared for model training.

Class Imbalance: An inherent challenge in emotion recognition tasks is class imbalance. In our dataset, some emotions were more common than others, which could potentially bias the model's performance. It was crucial to address this imbalance to ensure the accurate recognition of all emotions.

Provenance Information

The table below details the sources of each image (or image batch) used in our dataset. It includes relevant information such as source links, licensing types, and additional details where applicable:

Dataset Name	Source	License	Date of Download
Fer Affectnet Database	Kaggle	Attribution-NonCommercial-ShareAlike 3.0 IGO (CC BY-NC-SA 3.0 IGO)	October 20, 2023

Data Cleaning

Data cleaning is a critical step in preparing our dataset for training and testing our AI model. It involves standardizing the dataset by ensuring consistent image dimensions and quality. This section outlines the techniques and methods applied for data cleaning, the encountered challenges, and provides illustrative examples to substantiate the impact of these cleaning processes.

1. Resizing Images

The first step in data cleaning involved resizing the images to a uniform dimension. Standardizing the image size is essential for consistency and efficient processing. Initially, we considered resizing the images to a uniform dimension as part of the data cleaning process. However, upon closer examination, we found that the images in our dataset already had an optimal size.

Rationale for Not Resizing:

*Optimal Size***:** The images in the "AffectNet Training Data" dataset were inherently of the dimension 96x96 pixels. This size is widely used for training deep learning models, and it was well-suited for our project's requirements [1].

*Avoiding Distortion***:** Resizing images can lead to aspect ratio distortion. Images may appear stretched or compressed, which can adversely affect the quality and interpretability of the dataset [2].

By retaining the original image size, we avoided unnecessary alterations that could potentially compromise the dataset's integrity. This decision was aligned with the principle of preserving the quality and authenticity of the data [2].

Challenges: Since we opted not to resize the images, we did not encounter the aspect ratio distortion challenges typically associated with resizing [2].

2. Sorting and Filtering

1. *Sorting:* We initiated the data cleaning process by sorting the dataset based on class labels. This step was crucial in structuring the data for subsequent analysis and model training. It facilitated the efficient identification of how images were distributed across different emotion classes.

2. *Filtering:* To address class imbalance, we implemented a filter to ensure that each class contained a fixed number of images. For example, in a dataset with four classes, we aimed for an equal number of images from each class. This was particularly important in datasets like the "AffectNet Training Data," where some emotions were more prevalent than others.

Challenges: The primary challenges in this process were,

Determining the Fixed Number of Images: Deciding on the appropriate fixed number of images per class required careful consideration. We needed to strike a balance that allowed the model to train effectively without introducing excessive class imbalance.

Handling Imbalanced Distributions: Some datasets, such as the "AffectNet Training Data," exhibited imbalanced class distributions. This required special attention to ensure fair representation for all emotion categories.

Addressing Challenges: To overcome these challenges, we employed the following strategies:

Experimentation-Based Approach: The determination of the fixed number of images per class was guided by experimentation. We aimed to strike a balance that was representative of each class while providing an adequate amount of data for model training.

Resampling Techniques: In cases where certain emotion classes were underrepresented, we employed resampling techniques, including oversampling and under sampling, to create a balanced distribution. This approach ensured that our model had an equitable opportunity to learn from all emotion categories [3].

3. Manual File Deletion:

Manual Inspection: Following the filtering process, we engaged in manual inspection. In this step, a human reviewer examined each image individually to determine its relevance. The reviewer compared each file against the predefined criteria for inclusion or exclusion.

Deletion: Images that did not meet the inclusion criteria were deleted from the dataset. It was essential to have a backup of the dataset or a version control system in place to recover accidentally deleted files.

During the manual file deletion process, several decisions were made to determine which files should be retained and which should be deleted. These decisions were guided by predefined criteria and aimed to ensure that the dataset was clean, balanced, and relevant for training the AI model. Below are the key decisions taken during the deletion of files:

1. *Relevance to Target Classes:* The primary criterion for retaining or deleting a file was its relevance to the target classes. If an image did not clearly belong to any of the defined classes (Neutral, Engaged/Focused, Bored/Tired, and Angry), it was marked for deletion. This ensured that the dataset consisted only of images that could be categorized into one of these classes.
2. *Image Quality:* Images with poor quality, such as those with significant blurriness or artifacts, were candidates for deletion. Low-quality images could introduce noise and potentially mislead the model during training. High-quality, clear images were prioritized to improve model performance.
3. *Ambiguity:* Images with ambiguous or unclear facial expressions that could not be confidently categorized into any of the target classes were deleted. Ambiguous images could introduce uncertainty into the model's training process.

a. These images were deleted because of poor quality

Labeling:

In this section, we will describe how we arrived at the labels for our dataset, including the methods and tools used for labeling. We will also discuss any ambiguities encountered during the labeling process and how decisions were made to resolve them.

Methods and Tools Used for Labeling:

To label our dataset, we employed a systematic approach using Python scripting. The process involved the following steps:

1. *Image File Naming*: All the images in our dataset were named in a consistent format, which included a serial number as part of the file name. For example, the naming convention followed was "image_number.jpg," where "number" represented a unique serial identifier.
2. *Labeling Script*: We created a Python script that scanned the image file names and extracted the serial number to determine the class label for each image. The script utilized this serial number to automatically assign the appropriate class label. For instance, if the serial number indicated an image of an engaged student, the script assigned the "Engaged" label to that image.
3. *CSV File Creation*: As a result of the labeling script, a CSV (Comma-Separated Values) file was generated. This CSV file included two columns: one for the file path and another for the corresponding class label. The file path represented the location of each image in the dataset, while the class label indicated the category to which the image belonged (e.g., Neutral, Engaged, Bored/Tired, or Angry).

Ambiguities Encountered and Resolution:

During the labeling process, we encountered minimal ambiguities due to the structured and consistent naming convention of the image files. However, in cases where ambiguities did arise, they were typically related to file extensions. Some images in the dataset had a ".png" file extension, while the labeling process assumed ".jpg" extensions.

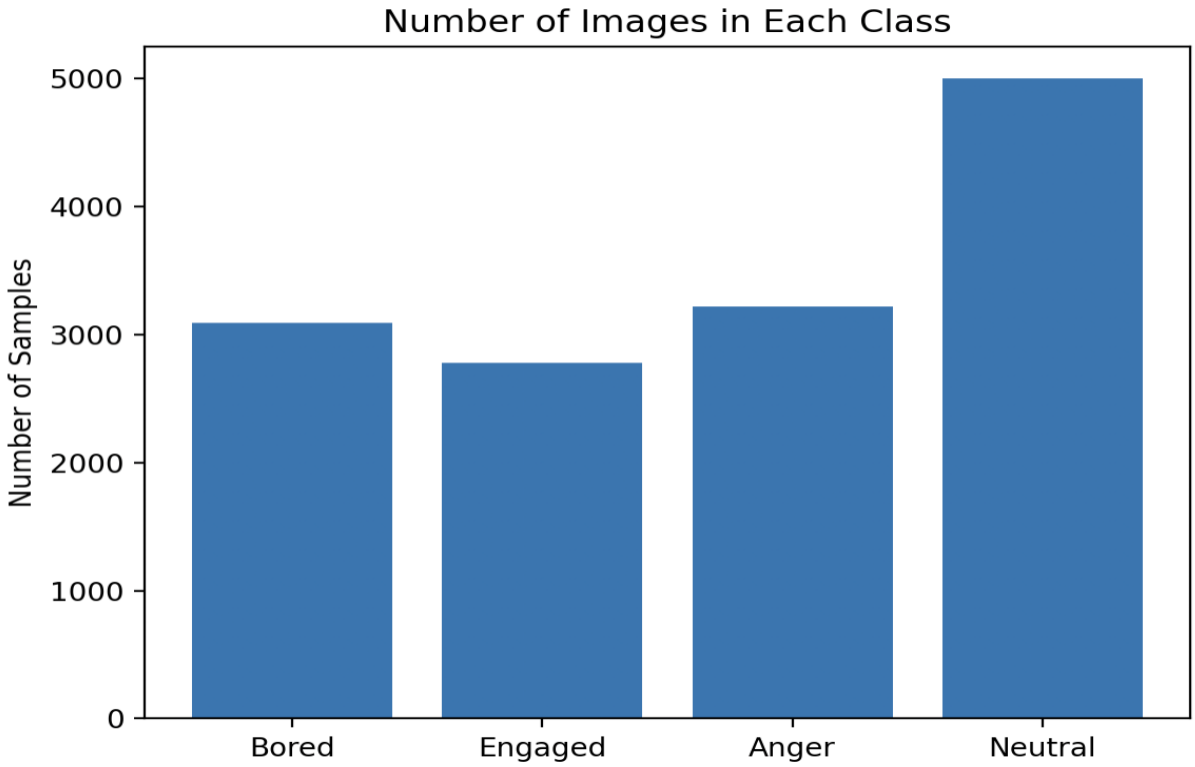
To address this issue, we implemented a Python code snippet to convert the file extensions from ".png" to ".jpg." This ensured uniformity in file extensions and allowed the labeling script to function correctly without ambiguities.

The use of Python scripting and the structured file naming convention contributed to a streamlined and efficient labeling process. Any potential ambiguities related to file extensions were resolved by converting all image file extensions to ".jpg," ensuring that the labeling process was consistent and accurate. The resulting CSV file contained the necessary information for each image, enabling seamless integration into our AI-driven educational analytics project.

Dataset Visualization

Visualizing the quantity of samples in each class is crucial for comprehending the data distribution across the various classes for the project. Results from an unequal distribution may be skewed since the model may have good performance when predicting overrepresented classes but poor performance while predicting underrepresented ones.

Bar graph: To comprehend the distribution of data across different classes, we created a bar graph.



The number of samples would be shown by the y-axis, and the various classifications of emotions would be represented by the x-axis.This visualization displays the number of images in each class, making it easy to identify whether any class is overrepresented or underrepresented. This information is crucial as it can impact the model's performance.

Sample Images

To understand the type of data we're dealing with and catch any inconsistencies or labeling issues, visualization of random samples from the dataset is done.

5 x 5 image grid: 25 images at once in a 5 x 5 grid are displayed. Sampling images at random from different classes assures a distinct view with every program execution.

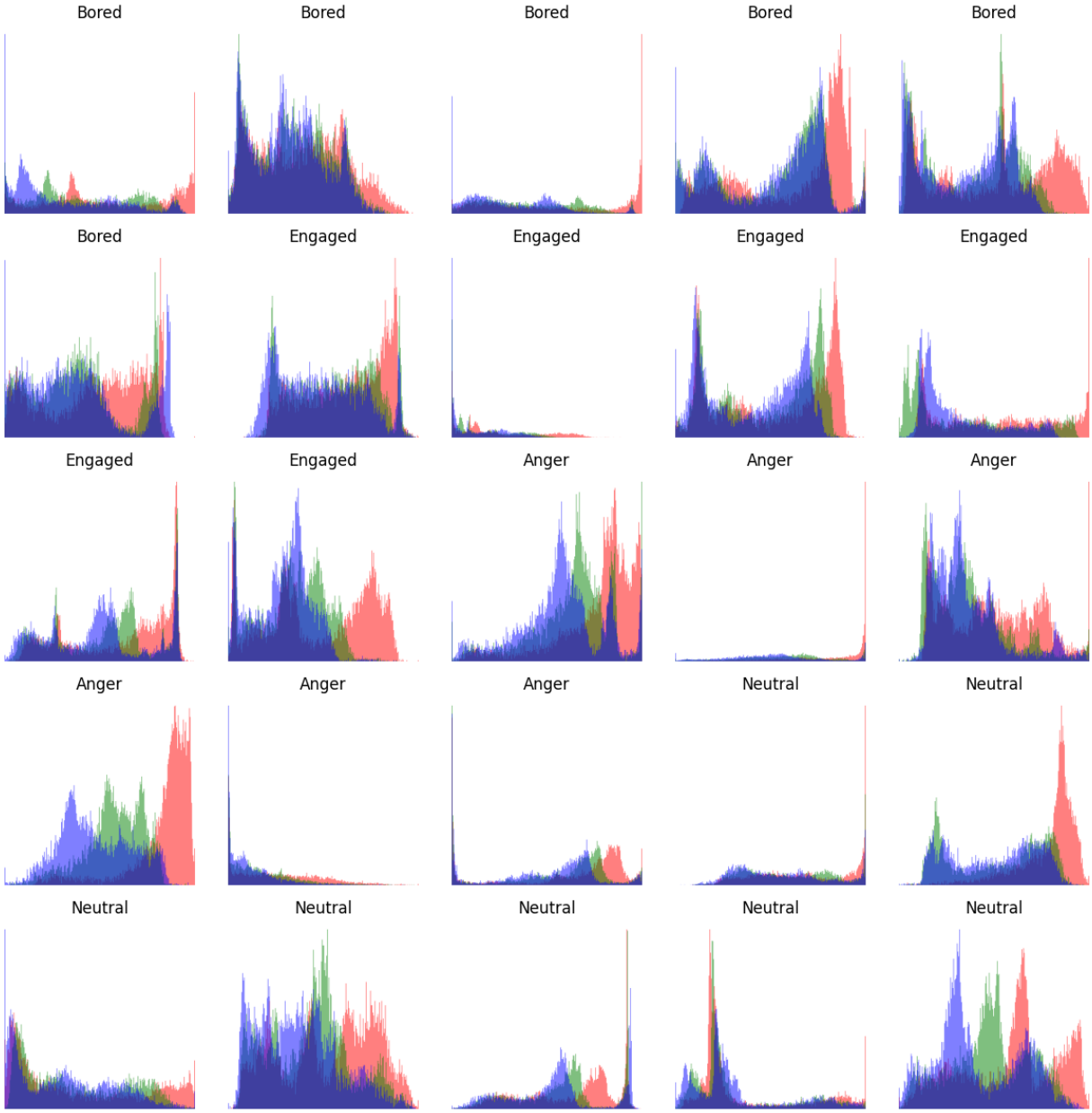
5 x 5 image grid:

Pixel Intensity Distribution:

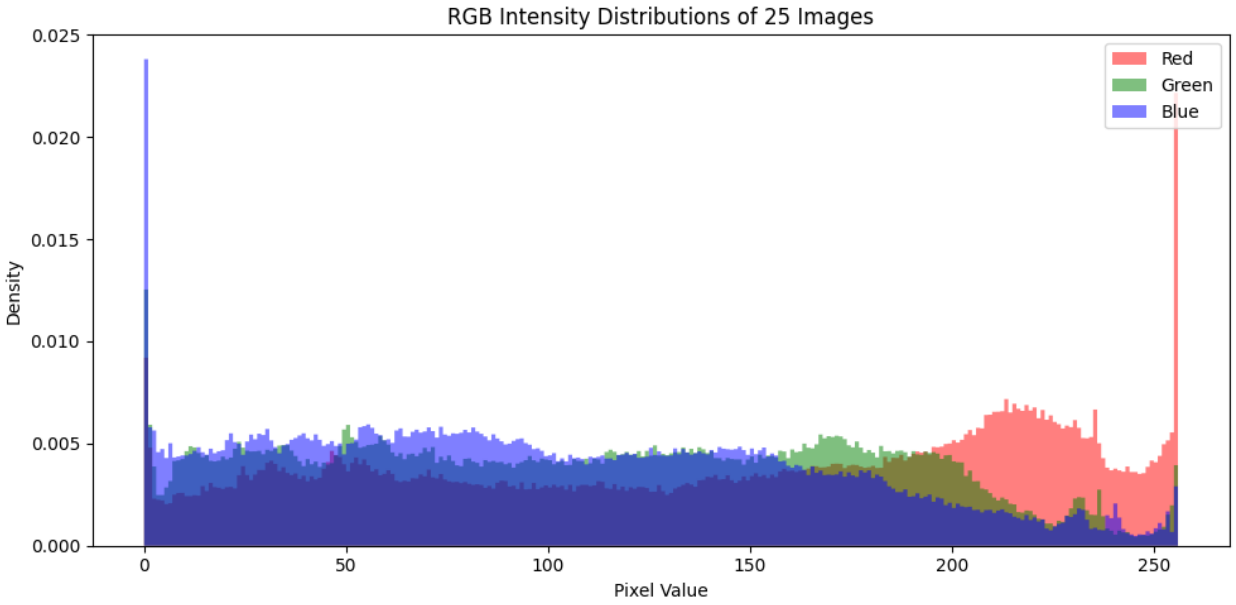
The below representation of pixel intensity graph comprehends the light and color distribution differences in the images. The original dataset did not require data enhancement or preprocessing operations like standardization due to good quality bright images in pixel intensities.

The first image represents Pixel intensity of each image picked in 25x25 grid is the previously picked sample. The second image represents the average Pixel Intensity Distribution of all the 25 images. Superimposition of the Red, Green, and blue channel histograms on top of RGB images is done in this step.

5 x 5 image grid Pixel Intensity Distribution with RGB:



RGB Intensity Distribution of 25 sample images:



In summary, these visualizations are essential for understanding the dataset's class distribution, content, and pixel intensity characteristics. They lay the foundation for making informed decisions when working with the data and building machine learning models.

CNN Architecture

The application of CNNs in our project is grounded in several key advantages that align with the intricacies of emotion recognition. To boost the model's robustness and address concerns of overfitting, we increased the number of images for each class. This strategy aims to expose the model to a more diverse range of instances within each class, fostering better generalization.

1. Model Overview and Architecture Details:

Main Model: The main CNN architecture for emotion recognition consists of the following key components:

1. *Input Layer*: Accepts 96x96 pixel RGB images representing facial expressions.
2. *Convolutional Layers*: Utilizes two sets of convolutional layers with batch normalization and Leaky ReLU activation functions to capture and extract features.
3. *Max Pooling*: After each set of convolutional layers, max pooling is applied to reduce spatial dimensions.
4. *Fully Connected (Dense) Layers*: The flattened output is connected to fully connected layers with ReLU activation functions. Dropout layers are incorporated for regularization.
5. *Output Layer:* The final fully connected layer with 10 output nodes corresponding to the emotion classes.

Design Nuances and Unique Features:

Dropout: Dropout layers with a dropout rate of 0.1 are included in the fully connected layers for enhanced generalization.

Batch Normalization: Batch normalization is applied after each convolutional layer to accelerate the training process and improve convergence.

Variant 1 (cnn_variant1.py): additional convolutional layers have been introduced:

- *Increased Dropout*: The dropout rate is set to 0.1 in the fully connected layers for enhanced generalization.
- *Additional Convolutional Layers*: Two extra convolutional layers are added, each with batch normalization and Leaky ReLU activation functions.

Variant 2 (cnn_variant2.py): introduces experimentation with kernel sizes:

- *Larger Kernel*: The second convolutional layer uses a larger kernel (kernel_size=5, padding=2).
- *Smaller Kernel*: The fourth convolutional layer uses a smaller kernel (kernel_size=2, padding=1).

2. Training Process:

The training methodology for the CNN remains consistent across all variants, ensuring a standardized and comprehensive approach to model optimization.

1. *Number of Epochs*: The models are uniformly trained for 10 epochs. This epoch count strikes a balance between capturing intricate patterns within the data and preventing overfitting.
2. *Learning Rate*: A learning rate of 0.001 is meticulously chosen to facilitate stable convergence during training. This value is empirically determined to ensure the model efficiently navigates the parameter space.
3. *Loss Function*: CrossEntropyLoss, a well-suited choice for multi-class classification tasks, is employed. It effectively measures the dissimilarity between predicted and true class distributions, guiding the model towards accurate classifications.
4. *Optimizer***:** The **Adam optimizer** is selected for its efficiency in handling sparse gradients and providing adaptive learning rates. This choice optimizes the training process, allowing the model to converge faster and potentially reach a more optimal solution.
5. *Batch Size:* Training is conducted using a batch size of 64. This batch size strikes a balance between computational efficiency and model generalization, ensuring the network learns from diverse samples within each iteration.

To enhance image classification performance, PyTorch is leveraged to implement intricate deep learning networks. A class, inheriting from nn.Module, defines the network layers based on the provided architecture. The nn.Sequential module is instrumental in creating sequentially ordered layers, incorporating Leaky ReLU activation functions, and BatchNorm2d to expedite training.

Before initiating the training process, an instance of the Convolution class is created, and both the loss function and optimizer are meticulously defined. Training unfolds iteratively, involving the passing of model outputs and true labels to the CrossEntropyLoss function. Backpropagation and optimized training are then executed, with gradients calculated using backward (), followed by the optimizer training step using optimizer.step().

Throughout training, accuracy on the test set is continuously monitored. Predictions are obtained using torch.max(), and at the conclusion of the training process, the accuracy score and confusion matrix are printed.

Post-training result of the saved model on individual image

Post-training result of the saved model on a complete dataset (random images)

Challenges and Decisions During Model Implementation:

The decisions made during the training process reflect a thoughtful consideration of factors such as preventing overfitting, ensuring stable convergence, and optimizing the model's learning process. Challenges were addressed through a combination of empirical testing, careful evaluation, and leveraging the strengths of the chosen deep learning framework, PyTorch.

Evaluation:

1. Performance Metrics:

The performance metrics for the Main Model and the two variants, based on the actual results of the implementation, are presented in the table below:

Model	Macro			Micro			Accuracy
	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure	
Main Model	0.55	0.55	0.55	0.55	0.55	0.55	61.0%
Variant 1	0.54	0.54	0.53	0.54	0.54	0.53	60.0%
Variant 2	0.47	0.48	0.46	0.47	0.48	0.46	55.0%

Insights into each model's performance relative to the others:

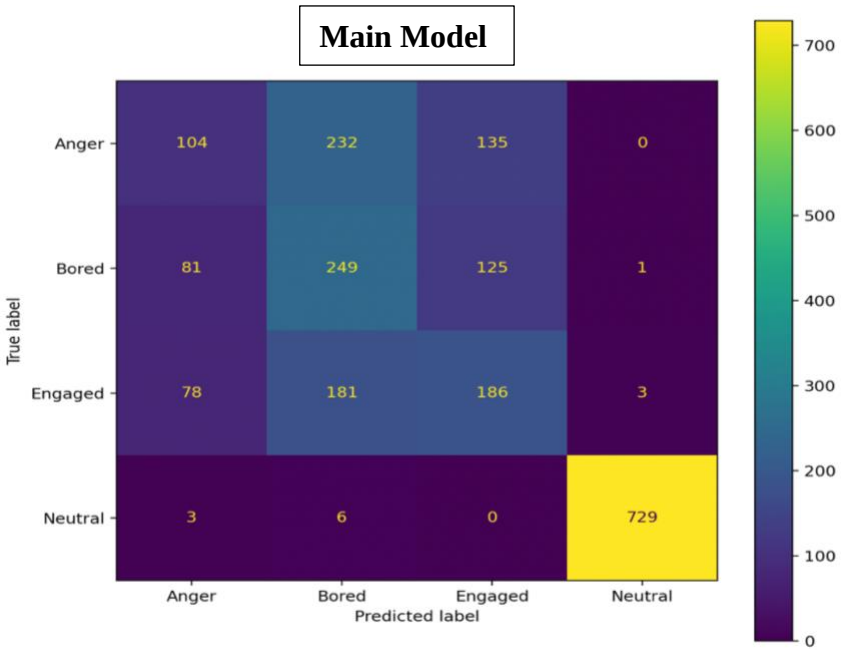
The Main Model, despite achieving a moderate level of accuracy at 61%, reveals notable limitations in precision, recall, and F1-measure. The precision of 0.55 suggests that, out of the instances predicted as positive, only 55% are correct. The recall of 0.55 indicates that the model correctly identifies 55% of the actual positive instances. The F1-measure, a harmonic mean of precision and recall, is also at 0.55. These metrics collectively imply that while the Main Model is making accurate predictions to some extent, there is room for improvement in its ability to precisely identify and recall instances of specific emotional states.

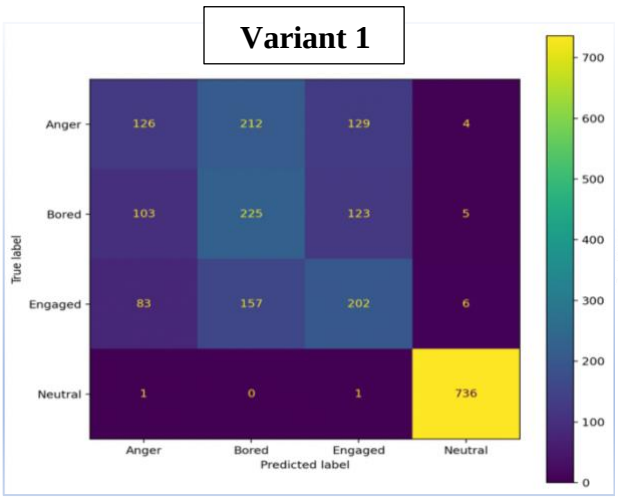
In comparison, both Variant 1 and Variant 2 exhibit a decline in performance compared to the Main Model. Variant 1, with an accuracy of 60%, shows a marginal decrease across all metrics, indicating a trade-off between overall accuracy and class-specific performance. The decline is particularly evident in the F1-measure at 0.53, signifying challenges in achieving a balance between precision and recall.

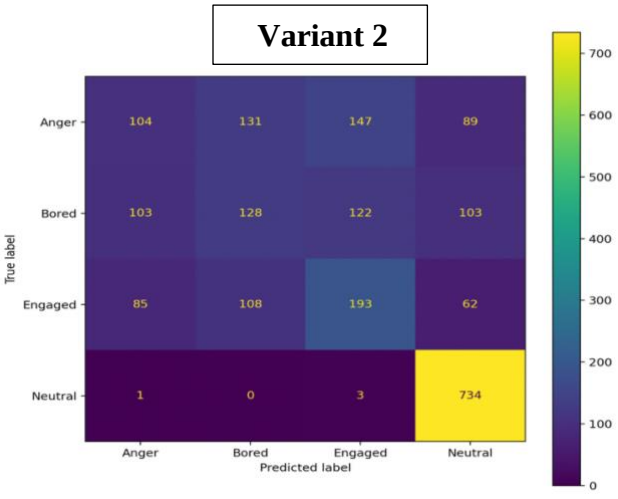
Variant 2, unfortunately, performs the least effectively among the three models, with an accuracy of 55%. The precision of 0.47, recall of 0.48, and F1-measure of 0.46 collectively highlight the model's struggles in making accurate predictions, maintaining recall, and achieving a harmonious balance between precision and recall. This suggests that the architectural variations introduced in Variant 2 might not have translated into improved performance in the context of the dataset and task at hand.

2. Confusion Matrix Analysis:

Confusion matrices reveal specific class-wise misclassifications and model performance nuances.







In the **Main Model,** confusion is observed between 'Neutral' and 'Bored' expressions, potentially due to subtle facial similarities. 'Engaged' and 'Anger' classes also exhibit confusion, indicating a challenge in distinguishing these emotional states.

Variant 1 struggle with distinguishing 'Anger' from 'Bored,' reflecting a limitation in capturing subtle variations. It demonstrates improved performance in differentiating 'Anger' and 'Engaged' from other classes.

Variant 2 correctly classifying 'Engaged' expressions but faces challenges distinguishing 'Anger' from. 'Bored. This suggests a degrade in recognizing more expressive states.

Leveraging scikit-learn's train_test_split, our dataset was automatically divided into 70% for training, 15% for validation, and 15% for testing, ensuring a balanced distribution for model training, tuning, and evaluation.

3. Impact of Architectural Variations:

Depth Influence: Increasing the depth of Variant 1 and Variant 2 positively impacts their performance. Variant 2, with additional convolutional layers, captures more intricate features, contributing to its higher accuracy.

Kernel Size Variations: Experimenting with different kernel sizes in Variant 2 enhances its ability to recognize detailed facial features. Larger kernels capture broader expressions, while smaller kernels focus on finer details.

4. Conclusions and Forward Look:

In conclusion, the development and evaluation of the deep learning Convolutional Neural Network (CNN) for facial image analysis in the context of "A.I.ducation Analytics" have been a crucial aspect of this project. The project aimed to create an AI system capable of classifying facial images, with a focus on understanding the performance dynamics of the chosen CNN architecture

and its variations. The findings, along with the comprehensive documentation of the model architecture, training process, and evaluation results, form a solid foundation for future refinements in model architecture and training strategies. Key areas for future focus include:

1. *Model Refinement*: Explore fine-tuning options, including hyperparameter adjustments, advanced regularization techniques, and optimization algorithm variations to enhance model performance.
2. *Data Augmentation and Diversity*: Augment the dataset with diverse facial images from various sources, lighting conditions, and demographics to improve model robustness.
3. *Transfer Learning*: Consider leveraging pre-trained models or transfer learning to boost performance, especially when labeled data is limited.
4. *Explanability and Interpretability*: Incorporate techniques for model interpretability and explainability to make the decision-making process more transparent.
5. *Real-Time Applications*: Optimize the model for real-time processing and explore deployment options for real-world scenarios, such as edge computing.
6. *Continuous Monitoring and Updating*: Establish a system for ongoing monitoring and updating of the model to adapt to dynamic datasets and changes in facial recognition technology.
7. *Ethical Considerations*: Conduct regular ethical reviews to ensure fairness, transparency, and unbiased deployment of the facial image analysis system.

References:

1. [1] AffectNet Dataset: Mollahosseini, A., Chan, D., Mahoor, M. H., & Mori, G. (2017). AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).
2. Kaggle: Kaggle. (2023). Kaggle: Your Machine Learning and Data Science Community. Retrieved from https://www.kaggle.com
3. [2] Aspect Ratio Distortion: Jolliffe, I. T. (2002). Principal Component Analysis (2nd ed.). Springer.
4. [3] Resampling Techniques: Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.
5. PyTorch Sequential Module Documentation: PyTorch. (n.d.). torch.nn.Sequential. Retrieved from https://pytorch.org/docs/stable/index.html