
# COMP 6721 - Applied Artificial Intelligence (Fall 2023)

## Project Assignment, Part 2

**Team Name: AK_17**

**Team Members:** **Alay Parikh (40269382)**

**Jenish Akhed (40270365)**

**Shruti Pavasiya (40270486)**

**Specializations:**

1) **Data Specialist: Shruti Pavasiya**
2) **Training Specialist: Jenish Akhed**
3) **Evaluation Specialist: Alay Parikh**

**GitHub Link: https://github.com/parikhalay/Emotion-Detection**

# Table of Contents

# Dataset

**Overview of the Dataset:**

The dataset used for this project is the "Affect Net Training Data" obtained from Kaggle[1], which contains a large collection of facial images categorized into different emotions.

Total number of images (including all classes) in existing dataset is: **28834**

There are 8 type of classes in the dataset, which are named below with the number of images in each class.

1) **Neutral Class:**
   Total Images: 5127
2) **Surprise Class:**
   Total Images: 4040
3) **Contempt Class:**
   Total Images: 2872
4) **Anger Class:**
   Total Images: 3219
5) **Disgust Class:**
   Total Images: 2478
6) **Sad Class:**
   Total Images: 3092
7) **Happy Class:**
   Total Images: 5045
8) **Fear Class:**
   Total Images: 2960

**Special Characteristics of the Dataset :**

1. Diverse Emotions: The dataset includes a wide range of emotions, making it suitable for training a model to recognize not only basic emotions but also nuanced expressions. Images are captured from all front angles.

2. Real-World Variety: The images in the dataset feature individuals from various demographics, making it applicable to real-world scenarios.

3. Varied Lighting and Backgrounds: The images may have diverse lighting conditions and backgrounds, which adds to the complexity of emotion recognition.

**Justification for Dataset Choices:**

1. **Relevance to the Project:** The dataset is highly relevant to the project's objective, which is to develop an AI system for analysing and categorizing student expressions in an academic setting. Emotion recognition is a key component of this system.
2. **Rich Emotion Coverage:** The dataset provides a comprehensive collection of emotions, enabling the model to recognize a broad spectrum of emotional states.
3. **Real-World Applicability:** The dataset's diversity in terms of subjects, expressions, and conditions aligns with the real-world scenarios in which the AI system will be deployed.

**Challenges:**

1. **Data Size:** The dataset is extensive, which can be both an advantage and a challenge. Training a model on a large dataset requires significant computational resources.
2. **Class Imbalance:** While the dataset covers a variety of emotions, there may be class imbalance issues, with some emotions having more examples than others.
3. **Data Pre-processing:** Due to the dataset's diversity, pre-processing steps such as resizing and data augmentation are required to ensure uniformity for model training.

**Provenance Information:**

The "AffectNet Training Data" dataset was sourced from Kaggle, and it is provided by Noam Segal. The dataset's original page on Kaggle contains information regarding its origin, licensing, and other relevant details. Below is a summary of the provenance information:

| Image Class | Licencing Type | Comments |
|---|---|---|
| Kaggle (Noam Segal) | Attribution-NonCommercial-ShareAlike 3.0 IGO (CC BY-NC-SA 3.0 IGO) | [Kaggle Dataset] |
| Mollahosseini et al., 2017 | N/A | Affect Net dataset |
| Affect Net Research Project (2017) | N/A | Original source of the dataset |

# Data Cleaning

## Techniques and Methodologies:

**File Format:** Conversion from .png file format to .jpg file format for uniform data visualization. Following are the files used to achieve that process:

- **pngTojpg.py:** This script is designed to convert PNG image files to JPG format. It iterates through directories, identifies PNG files, converts them to JPG, and replaces the original PNG files[2].

- **datasetShrinking.py:** This script selects and copies a specific number of random images from a source directory to a target directory. It's useful for creating smaller subsets of a dataset for testing and experimentation[2].

1. **Resizing Images:**
    - To standardize the dataset, the input images are resized to a consistent dimension. The target dimensions for resizing are set to a width of 100 pixels and a height of 100 pixels (target_width = 100 and target_height = 100).

    - The resizing process preserves the aspect ratio to ensure that the image's proportions remain intact. If the original width is greater than the original height, the width is set to the target width, and the height is calculated accordingly. Conversely, if the original height is greater, the height is set to the target height, and the width is calculated accordingly.

    - The interpolation method used for resizing is cv2.INTER_LANCZOS4, which is a high-quality interpolation method that helps avoid pixel loss and maintain image quality during resizing[3].

2. **Brightness Adjustment:**
    - A brightness adjustment is applied to the images. The brightness factor is set to 1.1 (brightness_factor = 1.1), which slightly increases the overall brightness of the image.

    - The adjustment is applied after resizing and pasting the image onto a canvas, ensuring that the entire image, including the canvas area, is adjusted.

**Challenges Encountered and Addressed:**

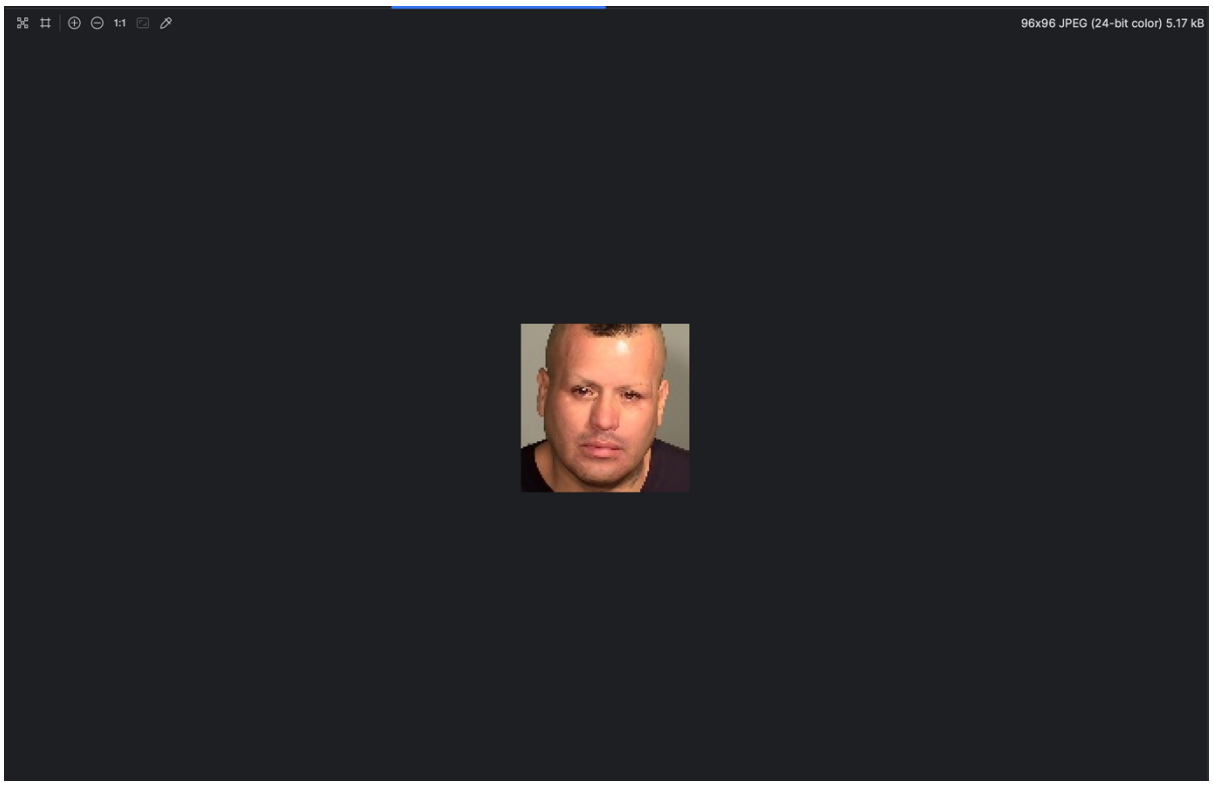
- **Aspect Ratio Preservation:** One challenge during resizing was preserving the aspect ratio of the original images. This was addressed by calculating the new dimensions based on the aspect ratio, ensuring that the images did not appear distorted.

- **Brightness Adjustment:** While applying brightness adjustments, it was important to ensure that the entire image, including the canvas area, was adjusted consistently. This was addressed by applying the brightness adjustment after the image was pasted onto the canvas.

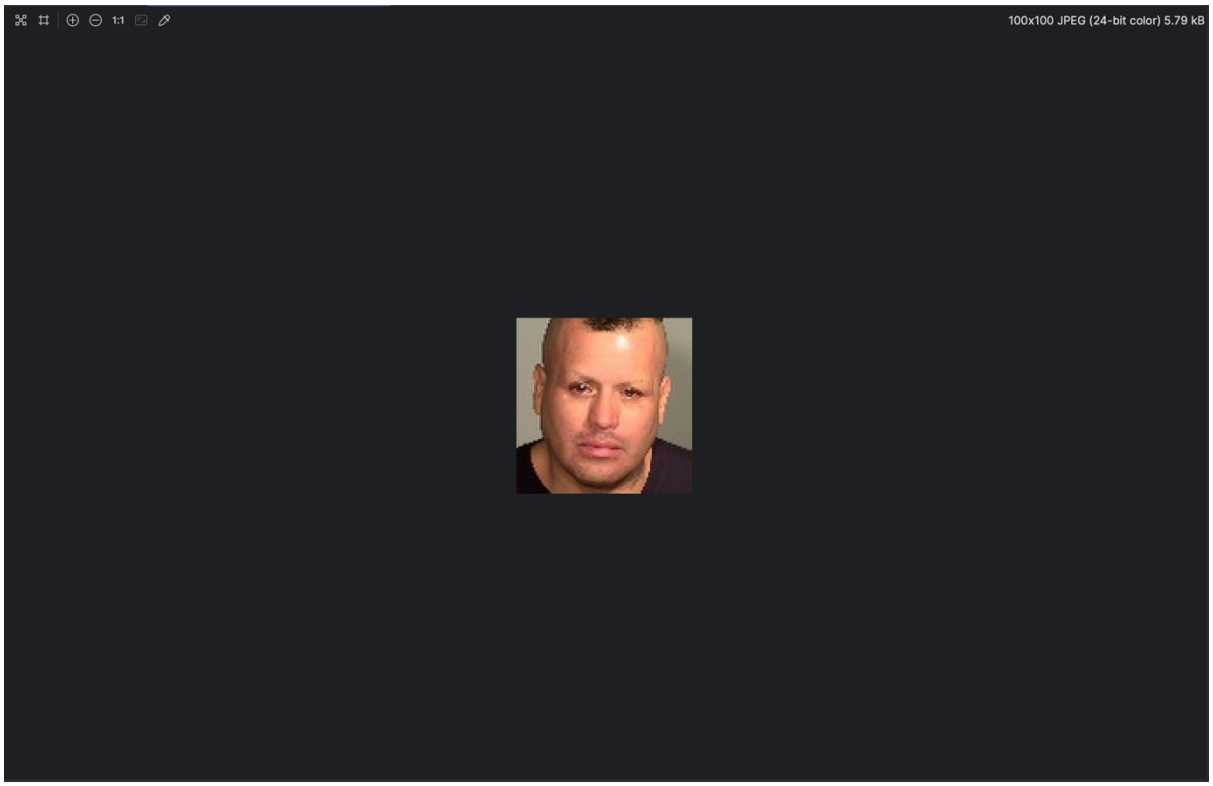
- **Pixel Loss During Resizing:** To prevent pixel loss and maintain image quality, the interpolation method **cv2.INTER_LANCZOS4** was chosen for resizing. This high-quality interpolation method results in smoother and sharper images after resizing.

Examples of the Before-and-After Cleaning Effect:

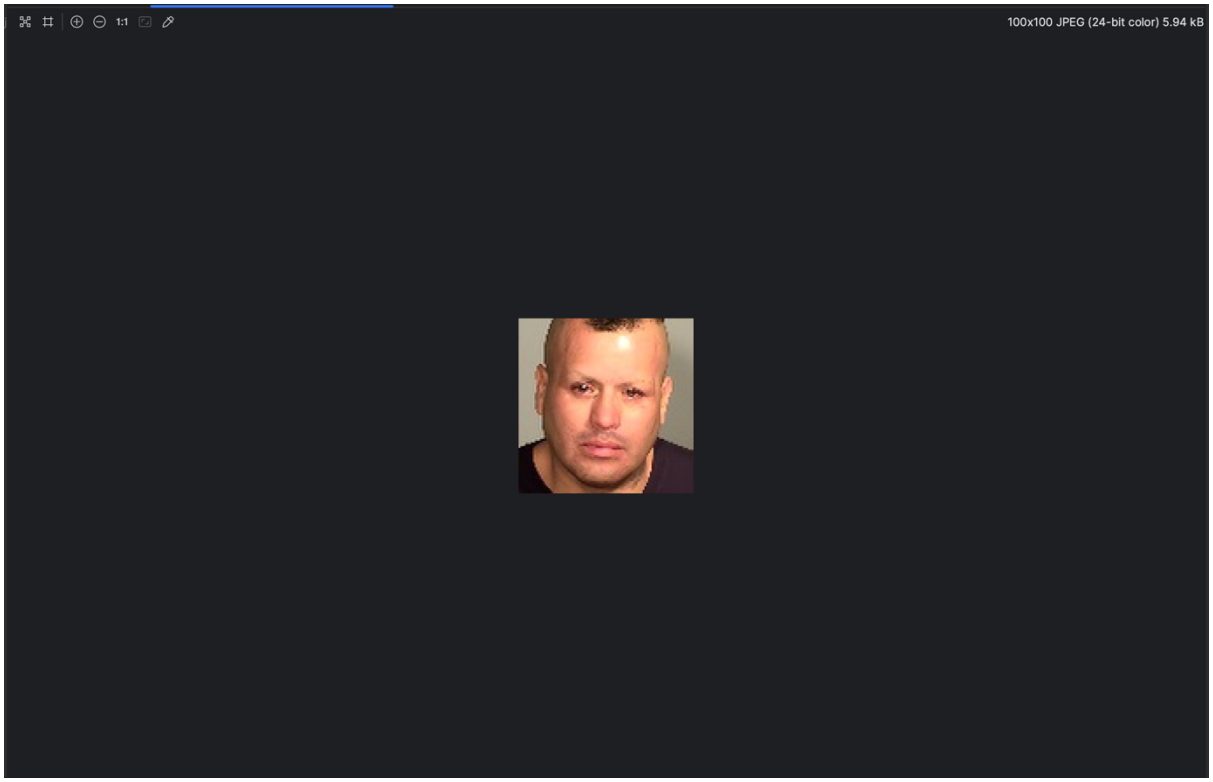
Original Image:




Resized Image with Aspect Ratio Preservation:



Resized Image with Brightness Adjustment:




In the examples, the original image is first resized while preserving its aspect ratio. The resized image is then pasted onto a canvas of the target dimensions. After pasting, a brightness adjustment is applied, enhancing the overall brightness of the image.

The data cleaning process ensures that all images are consistent in size and appearance, making them suitable for subsequent data analysis and model training.

# Labelling

Dataset labelled in reference dataset:

- The dataset was created on Label box[4], a data labelling platform.

- It is expected to be used for facial expression analysis, including emotions such as neutral, engaged, bored, angry, confused, and distracted.

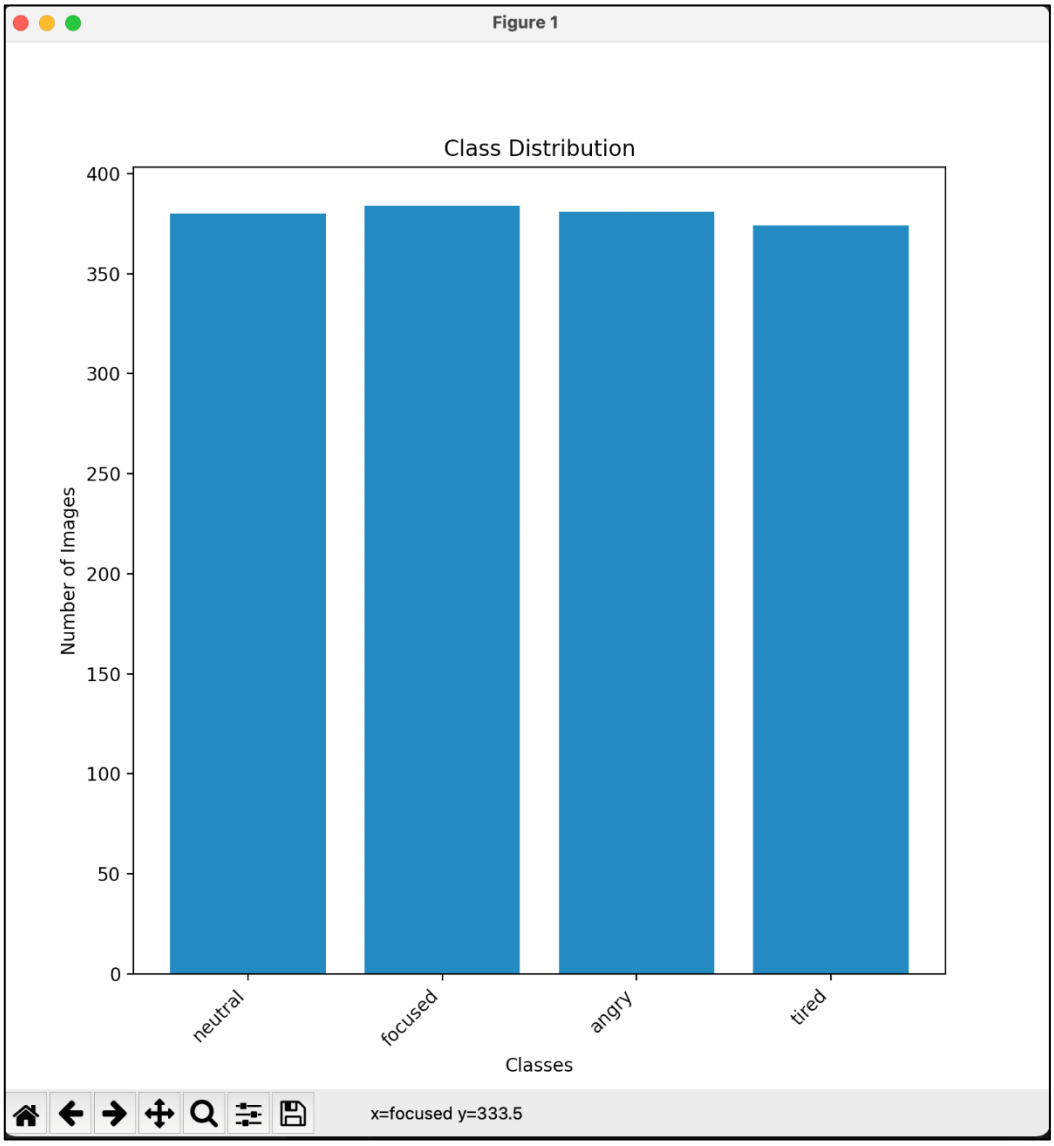
- The Affect Net dataset was already labelled into its respective classes, so there was no need for further labelling and analysing with label box as it is already labelled with label box tool.

# Data Visualization

## 1) Class Distribution(Bar Graph)[5]:

- The bar graph below shows the number of images in each class.
- It provides insights into whether any class is overrepresented or underrepresented.
- The class labels are listed on the x-axis, and the number of images is on the y-axis.

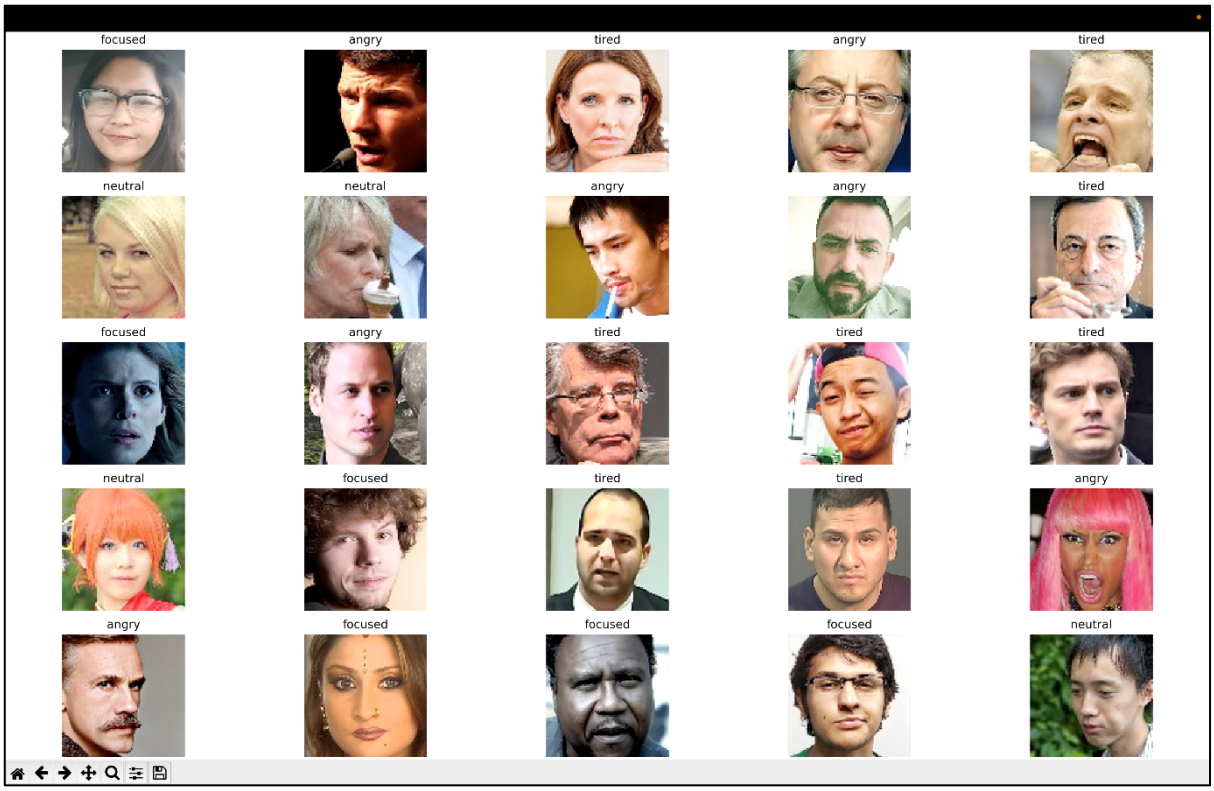
Example:



## 2) Sample Images(5x5 Grid)[6]:

- A collection of 25 random images is presented in a 5x5 grid.
- The images are randomly chosen from different classes upon every code execution.
- This visual representation helps understand the dataset's content and may help identify any noticeable anomalies or potential mislabelling's.

Example:




## 3) Pixel Intensity Distribution[6][7][8]:

- For the same random images, histograms showing the distribution of pixel intensities are plotted.
- In the histogram, the pixel intensity distribution of the Red, Green, and Blue channels is overlaid on a single histogram.
- This provides insights into variations in lighting conditions among the images.

Example:


focused
angry
tired
angry
tired
neutral
neutral
angry
angry
tired
focused
angry
tired
tired
tired
neutral
focused
tired
tired
angry
angry
focused
focused
focused
neutral
Frequency
Pixel Intensity

# CNN Architecture

## 1) Model Overview and Architecture Details [13]:

The main model architecture is defined in the FacialExpressionCNN class.

Here's an overview of the layers:

**Convolutional Layers:**

self.conv1: Convolutional layer with 3 input channels, 32 output channels, a kernel size of 3, and padding of 1.

self.conv2: Convolutional layer with 32 input channels, 64 output channels, a kernel size of 3, and padding of 1.

self.conv3: Convolutional layer with 64 input channels, 128 output channels, a kernel size of 3, and padding of 1.

**Fully Connected (Linear) Layers:**

self.fc1: Fully connected layer with a flattened input size (determined by the convolutional layers) and 256 output features.

self.fc2: Fully connected layer with 256 input features and 4 output features, representing the number of classes (assuming 4 classes for facial expressions).

**Activation Functions:**

ReLU (Rectified Linear Unit) activation functions are applied after each convolutional layer (F.relu).

The final layer uses the log-softmax activation (F.log_softmax) to obtain the class probabilities.

**Flatten Layer:**

Before entering the fully connected layers, the output is flattened using x.view(x.size(0), -1).

**Model Initialization and Size Calculation:**

The model is initialized with an input size of (3, 90, 90) representing the colour channels (RGB) and image dimensions. The size of the flattened features after the convolutional layers is calculated dynamically using a dummy input.

**Regularization:**

The model incorporates regularization techniques to prevent overfitting:

Weight decay (L2 regularization) is applied in the optimizer (torch.optim.Adam) with a weight decay parameter of 1e-5.

Dropout is applied with a probability of 0.5.

**Variant 1:**

1. **Additional Convolutional Layer**: A new convolutional layer (**self.conv4**) has been added to the **FacialExpressionCNN** class. This layer is a **nn.Conv2d** with 128 input channels and 256 output channels, using a kernel size of 3 and padding of 1.
2. **Modified Forward Pass**: The forward pass method (**forward**) includes an additional relu-maxpool operation for the new convolutional layer (**self.conv4**).
3. **Adjusted Flattened Size Calculation**: The **_get_conv_output_size** method has been modified to accommodate the new convolutional layer by adding the **more** parameter and incorporating additional operations if **more** is set to **True**.
4. **Model Saving Name**: The model is saved with a different filename to distinguish it from the main model.

**Variant 2:**

1. **Modified Kernel Size in Convolutional Layers**: In the **FacialExpressionCNN** class, the kernel size for the convolutional layers (**self.conv1**, **self.conv2**, **self.conv3**) has been changed to 5.
2. **Model Saving Name**: Similar to Variant 1, the model is saved under a different filename.

These changes in Variant 1 and Variant 2 represent different architectural choices for the neural network, potentially impacting its performance in terms of accuracy, training time, and ability to generalize. Variant 1 adds complexity with an additional convolutional layer, while Variant 2 experiments with larger kernel sizes for existing layers.

## 2) Training Process [11]:

**Number of Epochs:**

The number of epochs is set to 15 (epochs = 15). This means the entire dataset is passed through the neural network 15 times during training.

**Learning Rate:**

The learning rate is set to 0.001 (lr=0.001). This is the step size used by the optimizer to update the weights during training.

**Loss Function:**

The loss function used is negative log-likelihood loss (F.nll_loss) combined with log softmax activation in the last layer of the neural network. It is suitable for multi-class classification problems.

**Optimizer:**

The Adam optimizer is used with a learning rate of 0.001 and weight decay of 1e-5. Adam is an adaptive learning rate optimization algorithm that combines the benefits of AdaGrad and RMSProp. Weight decay is a form of L2 regularization.

**Learning Rate Scheduler:**

An optional learning rate scheduler is used (torch.optim.lr_scheduler.StepLR). It adjusts the learning rate during training. In this case, the learning rate is reduced by a factor of 0.1 after every 5 epochs.

**Mini-Batch Gradient Descent:**

Mini-batch gradient descent is utilized. The data is divided into batches, and the model's weights are updated based on the average gradient from each batch. This helps in faster convergence compared to full-batch gradient descent.

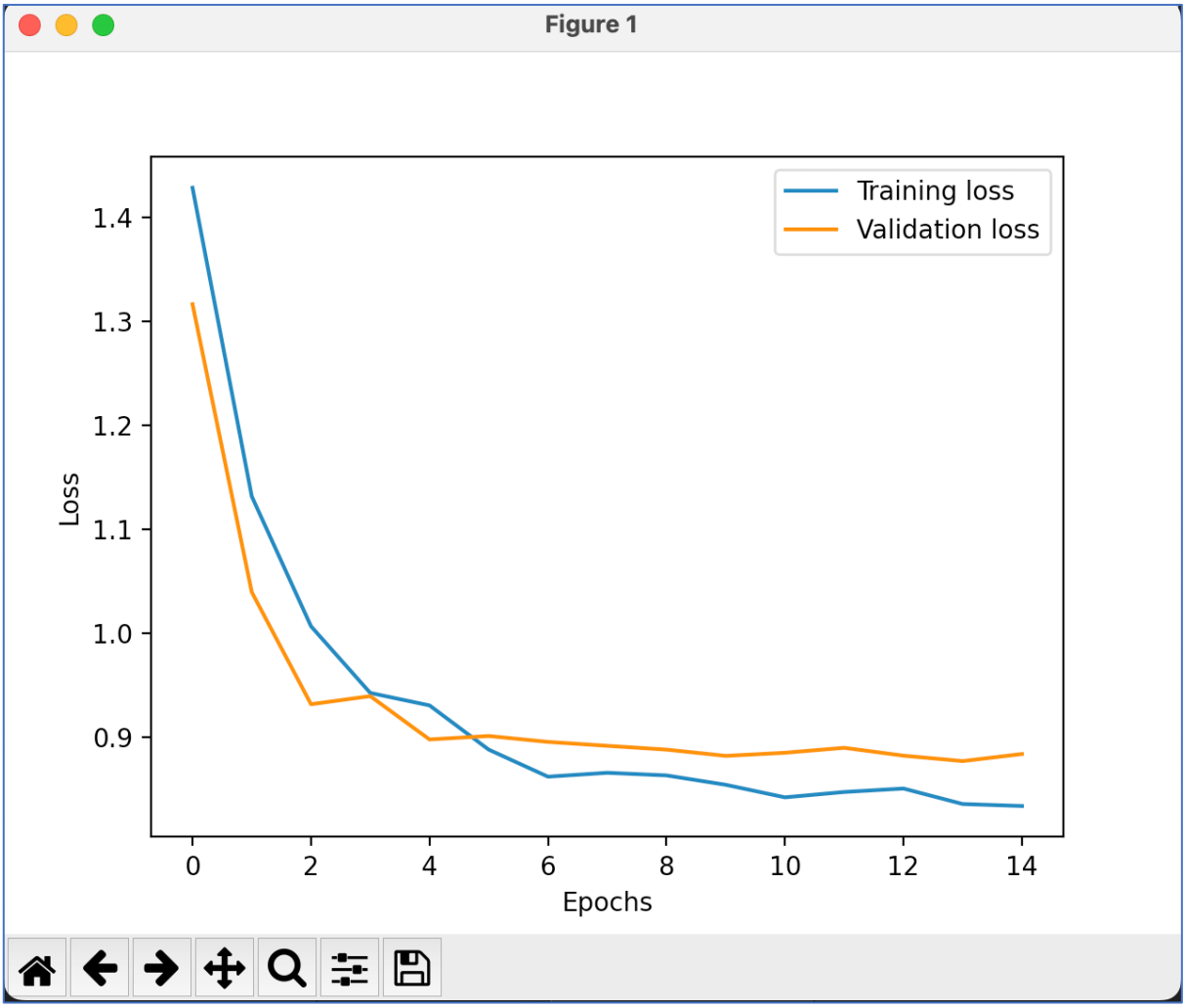
**Early Stopping:**

Early stopping is implemented using a custom EarlyStopping class. It monitors the validation loss, and if the validation loss does not improve for a certain number of consecutive epochs (controlled by the patience parameter), training is stopped early to prevent overfitting.

**Dropout:**

Dropout with a probability of 0.5 is applied after the first fully connected layer (self.dropout = nn.Dropout(0.5)). Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to zero during training.

In summary, the main model is trained using the Adam optimizer with a learning rate of 0.001 and weight decay for regularization. The learning rate is adjusted using a scheduler, and early stopping is employed to prevent overfitting. Mini-batch gradient descent is used for optimization, and dropout is applied for regularization. The training process is monitored for both training and validation losses, and the training loop is run for a maximum of 15 epochs with early stopping activated if the validation loss does not improve for 10 consecutive epochs.



# Evaluation

## 1) Performance Metrics [12]:

| Model | Macro | | | Micro | | | Accuracy |
|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | |
| Main Model | 0.5639 | 0.5657 | 0.5612 | 0.5784 | 0.5784 | 0.5784 | 57.84% |
| Variant 1 | 0.6299 | 0.6365 | 0.6325 | 0.6340 | 0.6340 | 0.6340 | 63.40% |
| Variant 2 | 0.5635 | 0.5506 | 0.5529 | 0.5686 | 0.5686 | 0.5686 | 56.86% |

**Main Model:**

- **Macro Average**: Precision, Recall, and F1-Score are balanced (around 0.56), indicating that the model performs uniformly across all classes on average.
- **Micro Average**: Slightly higher than the Macro average, suggesting better performance on larger classes.
- **Accuracy**: At 57.84%, it shows that more than half of the predictions are correct, but there is significant room for improvement.

**Variant 1:**

- **Macro Average**: Shows a notable improvement across all metrics compared to the Main Model, with values around 0.63, indicating a better balance between precision and recall across all classes.
- **Micro Average**: Also improved, which means that the model is not only doing better on average but also on the larger classes.
- **Accuracy**: The highest among the three models at 63.40%, suggesting this model is the most reliable for correct predictions overall.

**Variant 2:**

- **Macro Average**: Precision is similar to the Main Model, but Recall is slightly lower, suggesting that while it is as precise as the Main Model, it fails to identify as many relevant cases across the classes.

- **Micro Average**: Better than the Macro average, but lower than the Main Model and Variant 1, indicating it might not be performing as well on larger classes.
- **Accuracy**: The lowest at 56.86%, indicating that this model has the highest error rate in prediction among the three models.

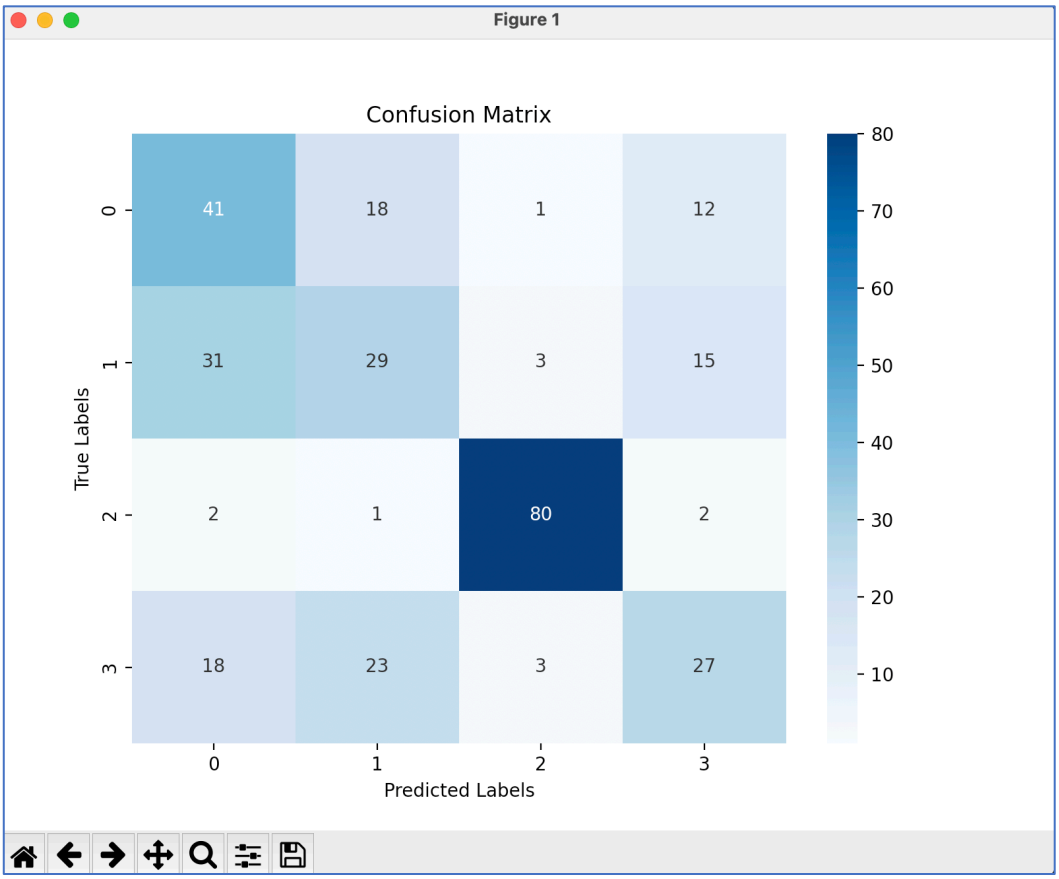
**Implications in the Context of Facial Image Analysis:**

- **Main Model**: This model could be considered a baseline; it offers a balance but no specific strength in precision or recall. In a facial image analysis context, this might mean it can be a general-purpose model but may not be ideal for situations where missing a positive (high recall) or ensuring a true positive (high precision) is crucial.
- **Variant 1**: With higher precision and recall, this model is more adept at correctly identifying the correct facial expressions without as many false positives or negatives. This could be more useful in applications where accuracy is critical, such as security systems or identifying subtle emotional cues.
- **Variant 2**: The lower recall relative to its precision suggests this model is conservative in predicting a facial expression; it prefers to be sure before making a prediction, leading to more false negatives. This could be useful in situations where false positives are a greater concern than missing out on potential true positives, possibly in sensitive applications where the cost of a false alert is high.

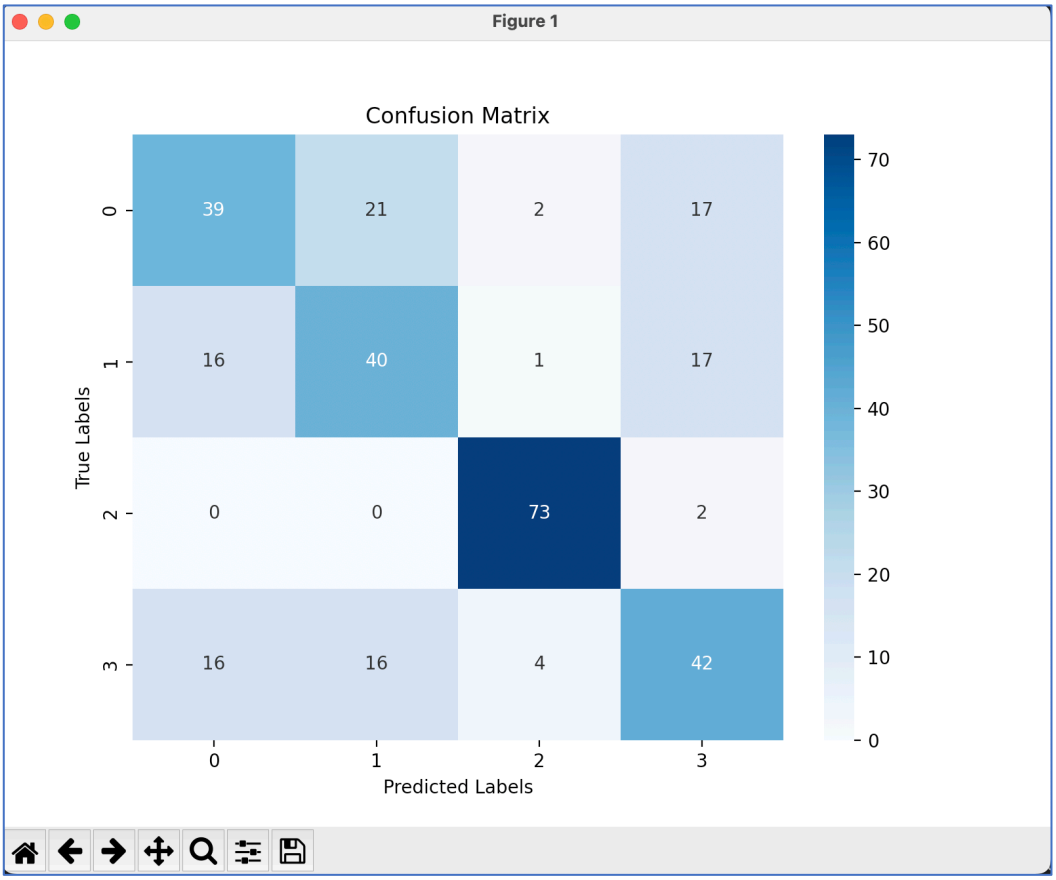
## 2) Confusion Matrix Analysis [12]:

In confusion matrix [0: angry, 1: focused, 2: neutral, 3: tired]

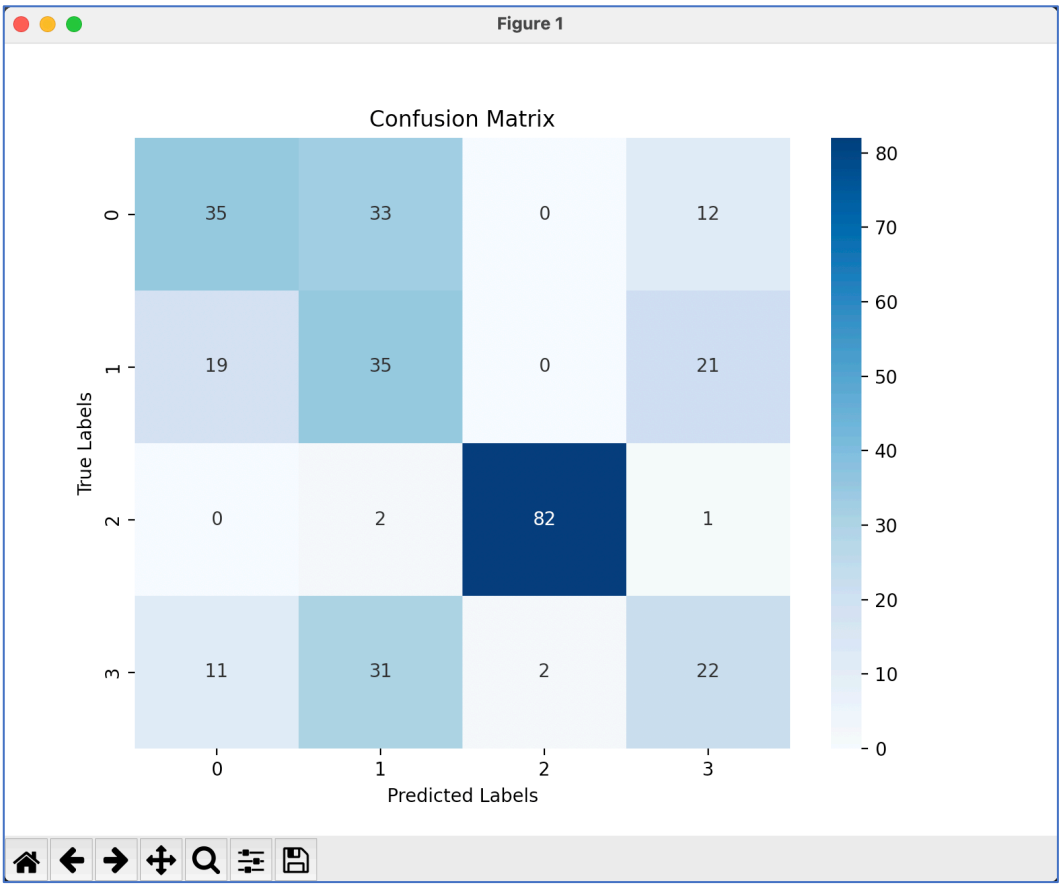
Main Model:




Variant 1:



Variant 2:




**Main Model:**

- **Most Frequently Confused**:
  - **Angry (0) with Focused (1)**: It's possible that the features of an angry face, such as furrowed brows, could be similar to a focused expression.
  - **Tired (3) with Focused (1)**: Tired and focused expressions might share certain features like drooping eyelids or a lack of smile, making them harder to distinguish.
- **Well-Recognized Class**:
  - **Neutral (2)**: This class is well recognized, which could be due to neutral expressions having less pronounced features, making them easier to distinguish from more expressive faces.

**Variant 1:**

- **Most Frequently Confused**:
    - **Angry (0) with Focused (1)**: The confusion here is slightly less than in the main model, suggesting an improvement in distinguishing between these emotions.
    - **Tired (3) with Focused (1)**: Still commonly confused, but there are improvements over the main model.
- **Well-Recognized Class**:
    - **Neutral (2)**: Continues to be recognized well, reinforcing the idea that the model is adept at identifying less expressive or more subdued facial features.

**Variant 2:**

- **Most Frequently Confused**:
    - **Angry (0) with Focused (1)**: The confusion persists, indicating that this is a systematic issue across models.
    - **Tired (3) with Focused (1)**: Again, these emotions are confused, perhaps due to similarities in facial cues.
- **Well-Recognized Class**:
    - **Neutral (2)**: This emotion is consistently recognized with high accuracy, suggesting that neutral expressions are the easiest for these models to identify.

**Dataset or Model-Specific Reasons for Misclassifications:**

- **Similar Expressive Features**: Angry and focused expressions may share intense features, such as narrowed eyes or furrowed brows. Tired and focused expressions might also share features like less active facial muscles or closed eyes.
- **Data Representation**: If the dataset contains more subtle differences between angry, focused, and tired expressions, the models may not capture these nuances effectively.
- **Model Generalization**: The models might not generalize well enough to distinguish between similar expressions, which could be addressed by increasing model complexity or training on a more diverse dataset.

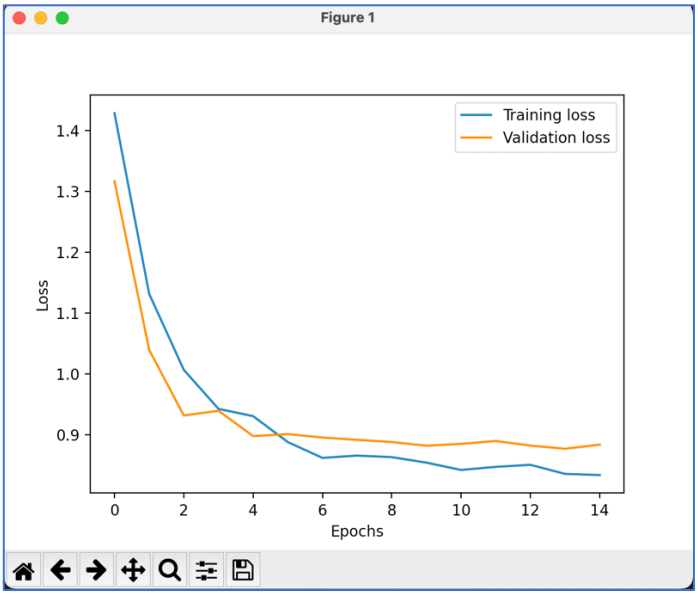
**Reasons Behind Well-Recognized Neutral Class's Success:**

- **Clear Distinction**: Neutral expressions likely have a clear distinction from other expressions, lacking the pronounced features of emotions like anger or tiredness.
- **Uniformity Across Data**: Neutral expressions might be more consistent across different subjects in the dataset, allowing the models to learn a more uniform representation.
- **Architectural Suitability**: Variant 1's architecture, in particular, seems to capture the features of neutral expressions effectively, leading to better performance.

## 3) Impact of Architectural Variations [8]:

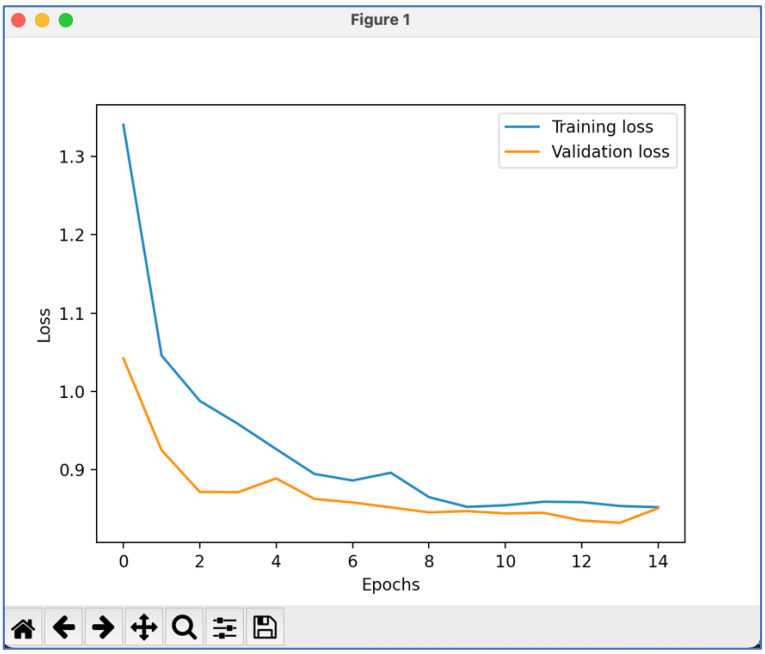
♦ How depth (number of convolutional layers) influenced performance:

**Main Model Performance:**




- The main model's performance gradually improved over epochs, as indicated by the decreasing validation loss.
- The model did not overfit, as shown by the continued reduction in validation loss over time.
- However, the model's final accuracy was 57.84%, which is respectable but suggests there is room for improvement.

**Variant 1 Performance:**




- Variant 1, with an extra convolutional layer, shows a more significant drop in validation loss earlier in training, indicating it might be capturing more complex features.
- The validation loss decreases and flattens out, suggesting that the model is not overfitting but rather stabilizing in its predictions.
- The final accuracy of Variant 1 is higher at 63.40%, which is a notable improvement over the main model.

**Implications of Depth Increase:**

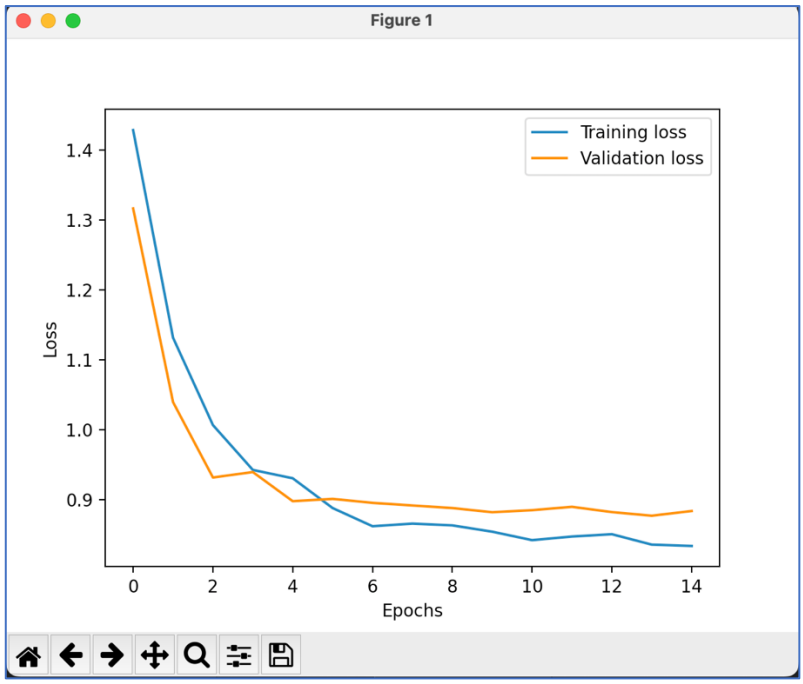
- The increase in depth for Variant 1 appears to have enabled the model to capture more detailed features of the data, contributing to better performance metrics.
- There is no clear indication of overfitting in Variant 1, given that the validation loss continues to decrease alongside the training loss. This suggests that the additional layer helped the model generalize better rather than memorize the training data.
- The increase in depth has to be handled carefully; more layers can sometimes lead to overfitting, but in this case, the model architecture and the data seem to have been well-matched, leading to an increase in generalization capability.

**Conclusions:**

- Depth in convolutional neural networks can indeed lead to a more nuanced understanding of the data, allowing the model to identify and utilize more complex patterns for classification.
- Variant 1 shows that additional depth, when properly regulated with techniques like early stopping, can lead to better performance without overfitting.
- It's also important to consider other factors that could contribute to improved performance, such as increased model capacity, more effective feature extraction, and potentially better handling of non-linear relationships within the data.
- While deeper networks have the potential for better performance, they require careful tuning to avoid overfitting. Monitoring validation loss is crucial to ensure that the model continues to generalize well to unseen data.

♦ How kernel size variations affected the model's recognition abilities:

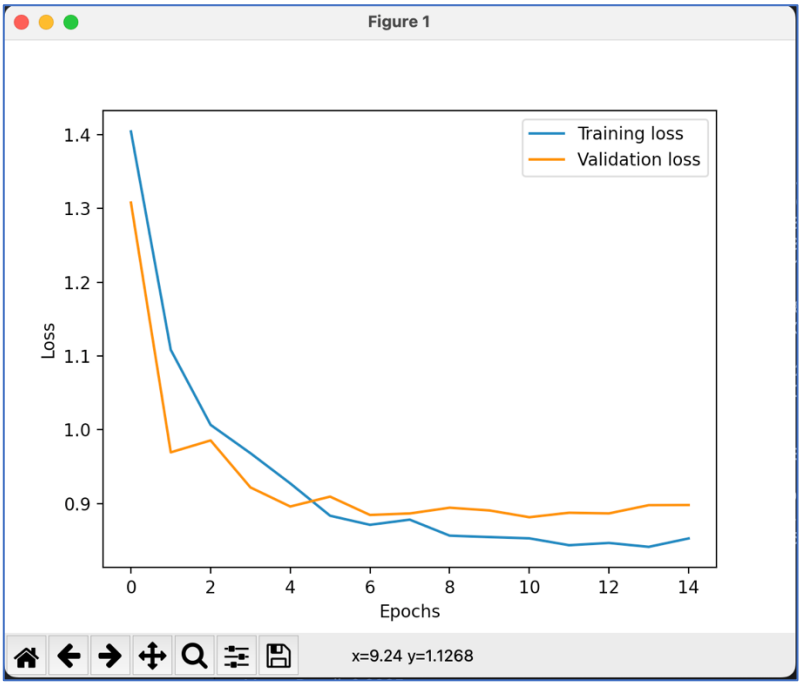
**Main Model:**




- The main model likely uses smaller kernels, which are typical for many standard CNNs. Smaller kernels are good at detecting finer details in

images since they look at fewer pixels at a time. This can be especially important for facial feature analysis, where subtle features like small changes in expression around the eyes or mouth can be crucial.

**Variant 2:**




- Variant 2 uses larger kernel sizes, as indicated by the information provided. Larger kernels consider a broader area of the input image at once, which can be beneficial for recognizing larger and more general features. However, they might skip over the finer details that smaller kernels would capture.
- Larger kernels reduce the spatial dimension of the feature map more drastically, leading to a more abstract representation of the input. This can sometimes lead to a loss of information about small but important features, which might explain the slightly lower performance metrics of Variant 2 compared to the main model.

**Recognition Abilities:**

- **Finer Features**: Smaller kernels can detect finer features, such as the subtleties of facial expressions. For instance, detecting the slight frown that differentiates a neutral face from a slightly angry one requires attention to fine detail that a small kernel could provide.
- **Broader Features**: Larger kernels are better at capturing broader features, like the general shape of the face or the hairline. However,

when it comes to facial expressions, which often rely on subtle shifts in small regions of the face, larger kernels might miss critical information.

**Reflection on Variant 2's Performance:**

- The slightly lower performance of Variant 2 might suggest that the larger kernel size did not capture the nuances of facial expressions as effectively as smaller kernels could. This can lead to a poorer differentiation between expressions, potentially explaining the lower recall and F1 score.
- In the context of facial expression analysis, where subtle features can significantly change the meaning of an expression, finer-grained feature detection is usually more beneficial.
- It's also possible that the larger kernels are too coarse for the scale of important features in the facial expression dataset, leading to a loss of critical information necessary for accurate classification.

**Conclusion:**

- The variations in kernel size between the main model and Variant 2 led to differences in the models' ability to recognize fine versus broad features. In the context of facial expression recognition, where detailed features are important, smaller kernels might be more appropriate. The results suggest that the main model, likely with its smaller kernels, was better suited to this task than Variant 2, which had larger kernels that may have missed some of the finer details important for distinguishing between different facial expressions.

## 4) Conclusions and Forward Look [8]:

Variant 1 outperformed the main model and Variant 2 in terms of accuracy, precision, recall, and F1 score. The performance gains in Variant 1 can likely be attributed to the additional convolutional layer it possesses, which allowed it to capture more complex features in the data. Despite having an additional layer, which generally increases the risk of overfitting, Variant 1 maintained a steady decrease in validation loss, suggesting that the model was learning generalizable patterns rather than memorizing the training data.

The main model showed decent performance but was outshined by Variant 1, which indicates that the depth of the model was a limiting factor.

Variant 2, which experimented with larger kernel sizes, did not perform as well as Variant 1. The larger kernels might have caused the model to overlook finer details in the facial expressions, which are crucial for accurate classification in tasks such as facial expression recognition.

**Suggestions for Future Refinements:**

1. **Model Architecture Refinements:**
    - Experiment with a **hybrid approach** that uses both small and large kernels in different layers to capture a wide range of features from fine to coarse details.
2. **Training Strategies:**
    - Use **data augmentation** to increase the diversity of the training data, helping the model generalize better to new, unseen data.
    - Apply **transfer learning** by using a pre-trained network on a large image dataset and fine-tuning it on the specific facial expression dataset to take advantage of learned feature detectors.
    - Implement **regularization techniques** such as dropout, weight decay, or batch normalization to prevent overfitting, especially if the network architecture becomes deeper.
3. **Hyperparameter Tuning:**
    - Experiment with different **activation functions** that might capture nonlinearities in the data more effectively.
4. **Evaluation and Experimentation:**
    - Use **k-fold cross-validation** to ensure the model's robustness and reliability across different subsets of the data.

# References


[1] Dataset: https://www.kaggle.com/datasets/noamsegal/affectnet-training-data

[2] For OS related code: https://docs.python.org/3/library/os.html

[3] Resize Image: https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html

[4] Label Box - https://labelbox.com/

[5] Bar Graph: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html

[6] Subplots: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

[7] Histogram Calculator: https://docs.opencv.org/3.4/d8/dbc/tutorial_histogram_calculation.html

[8] ChatGPT: https://chat.openai.com/

[9] Image Histogram Checker: https://www.sisik.eu/hist

[10] Python: https://www.python.org/

[11] Pytorch: https://pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/

[12] Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[13] Convolutional Neural Network: https://en.wikipedia.org/wiki/Convolutional_neural_network