Real-time Facial Expression Recognition Using CNNs

Elif Gürkan 21946223
Tuna Özcan 21987058
Baver Kaçar 2200356840

04.06.2024

1 Introduction

Our project ambitiously addresses the complex challenge of real-time facial expression recognition utilizing advanced Convolutional Neural Networks (CNNs). The overarching goal is to develop a sophisticated system capable of accurately processing and interpreting human emotions from live video feeds instantaneously. This technology has substantial potential to revolutionize digital communications by making interactions more intuitive and emotionally resonant.

2 Method

The approach for developing a real-time facial expression recognition system leverages a carefully designed Convolutional Neural Network (CNN) architecture and employs several data processing and augmentation techniques to enhance the model's accuracy and performance. Below is a detailed explanation of the methods used, consistent with the provided code.

2.1 Data Preparation

The images are preprocessed to fit the input requirements of the CNN:

- **Image Resizing and Normalization:** Each image in the dataset is resized to a uniform 48x48 pixels and converted to grayscale to reduce computational complexity. The pixel values are then normalized to the range [0,1] by dividing by 255, which helps in faster convergence during training.
- **Data Augmentation:** To address class imbalance and increase the robustness of the model, real-time data augmentation is employed using Keras's ImageDataGenerator. This includes random transformations like

rotations, width and height shifts, shear intensity changes, zooming, and horizontal flipping. These augmentations help the model generalize better to new, unseen data by simulating a variety of facial orientations and expressions.

2.2 CNN Architecture

The CNN model is structured to effectively learn distinctive features from facial expressions. The architecture includes:

- **Convolutional Layers:** Several layers, each followed by a batch normalization layer and a ReLU activation function, designed to extract spatial hierarchies of features.
- **Pooling and Dropout Layers:** Max pooling layers reduce spatial dimensions, while dropout layers help prevent overfitting.
- **Global Average Pooling and Dense Layers:** A global average pooling layer reduces model parameters, followed by dense layers with L2 regularization for classification.
- **Output Layer:** A dense layer with softmax activation outputs the probability distribution over the seven emotions.

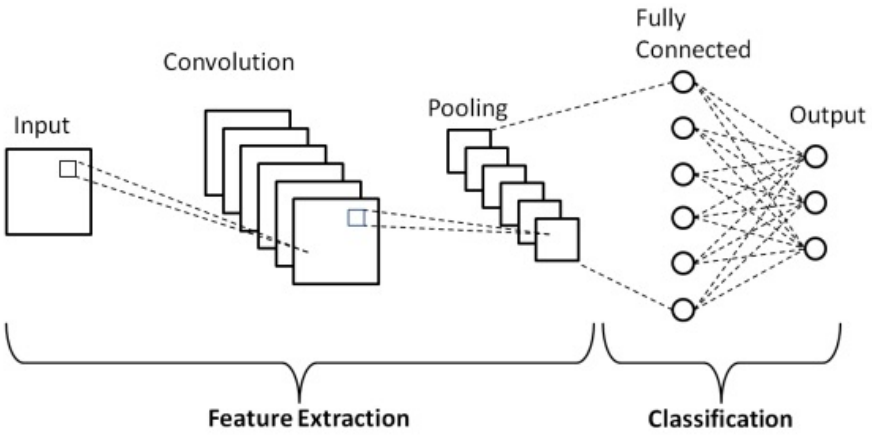


Figure 1: CNN Architecture

2.3 Training Procedure

- **Optimization and Loss Function:** The model uses the Adam optimizer and sparse categorical crossentropy as the loss function.

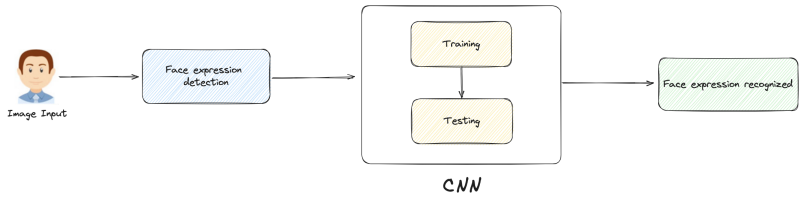


Figure 2: Overall Diagram

- **Callbacks:** EarlyStopping and ReduceLROnPlateau are used to enhance training efficiency.
- **Class Weighting:** Weights are assigned inversely proportional to class frequency to handle class imbalance.

3 Experimental Settings

3.1 Dataset Details

The model is trained and validated on a widely used facial expression recognition dataset from Kaggle, containing over 35,000 grayscale images depicting seven emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. Each image is standardized to 48x48 pixels. The dataset is divided into approximately 28,000 images for training, 3,500 for validation, and 3,500 for testing to ensure robust model evaluation. Data augmentation techniques like random rotations, shifts, and horizontal flips are applied to simulate real-world conditions and improve generalization. Images are normalized to a range of 0 to 1 to enhance learning efficiency.
In our project, it is not necessary to use video data. The dataset we have is sufficient for facial expression recognition. By using data augmentation techniques (such as random rotations, shifts, and horizontal flips), we can expose our model to conditions similar to those in the real world. These techniques improve the model's generalization ability, allowing it to perform well without video data. Our model can be designed to perform real-time facial expression recognition without video data. We can develop an architecture that handles high-resolution video streams and recognizes facial expressions instantly.

3.2 Computational Environment

Training and evaluation are conducted on a high-performance computing environment equipped with GPUs to expedite the training process. We utilize TensorFlow and Keras libraries for building and training the model, leveraging their robust, scalable frameworks suited for deep learning tasks. The specific use of a single GPU configuration is enforced to avoid unintended resource contention which can lead to suboptimal training performance.

3.3 Evaluation Metrics

The model's performance is assessed using the following metrics:

- **Accuracy**: Measures the proportion of correctly predicted instances over all predictions.
- **F1-Score**: Harmonic mean of precision and recall, providing a balance between the two in cases of class imbalance.

4 Experimental Results

4.1 Distribution of Emotion Classes

The bar chart below displays the distribution of emotion classes within the dataset. Each bar represents the number of samples for a particular emotion class, providing insight into the class balance of the dataset.

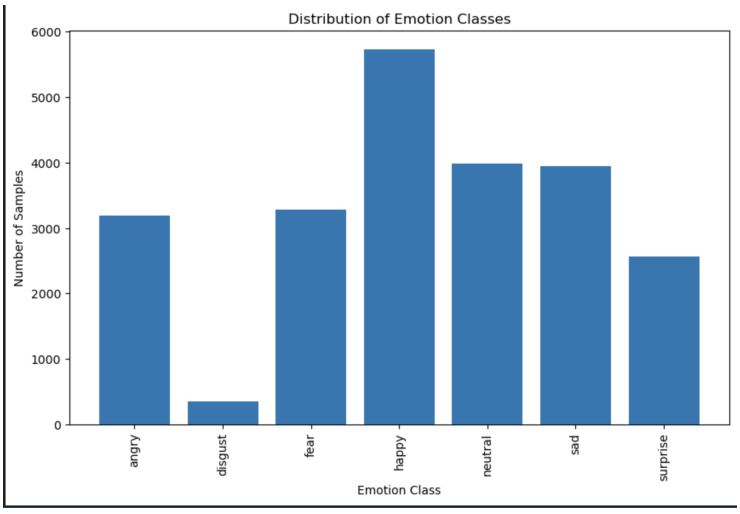


Figure 3: Distribution of Emotion Classes

- **Angry:** Approximately 3,000 samples.

- **Disgust:** Significantly underrepresented with fewer than 1,000 samples, highlighting a potential class imbalance issue.
- **Fear:** Around 4,000 samples, indicating a moderate representation.
- **Happy:** The most represented class with nearly 6,000 samples, showing a strong presence in the dataset.
- **Neutral:** Approximately 4,500 samples, indicating a good representation.
- **Sad:** Around 4,000 samples, showing a moderate presence.
- **Surprise:** Less than 2,000 samples, suggesting another potential class imbalance.

4.1.1 Implications

The dataset exhibits class imbalance, particularly with the "Disgust" and "Surprise" classes having significantly fewer samples compared to others like "Happy" and "Neutral". This imbalance can affect the model's performance, leading to bias towards more frequently occurring classes. To mitigate this issue, techniques such as data augmentation for underrepresented classes or applying class weighting during model training can be employed.

4.2 Hyperparameters and Comparisons

4.2.1 Hyperparameter Selection and Rationale

- **1.Data Augmentation Parameters:**

  Rationale: Due to the class imbalance observed in the data, augmentation techniques were crucial. Augmentation increases the model's exposure to varied data scenarios, which is expected to improve generalizability.

- **Parameters Used:**

  Rotation: 20 degrees to mimic slight head movements.

  Width and Height Shifts: 20

  Shear Intensity: 20

  Zoom: 20

  Horizontal Flip: Yes, because facial expressions can occur on either side of the face.

  Fill Mode: 'Nearest' to handle new pixel values created by transformations like shifts or rotations.

- **2. Class Weights:**

  Rationale: To address the significant class imbalance, weights were inversely assigned based on class frequency. This ensures that minority classes contribute more to the loss, pushing the model to pay more attention to them.

  Method: Class weights calculated as the ratio of the maximum augmentation factor to the specific class's augmentation factor.

- **3. Model Complexity and Regularization:**

  Rationale: The model includes several layers with dropout and batch normalization to prevent overfitting.

- **Parameters Used:**

  Dropout Rate: 30

  L2 Regularization: Applied in the dense layers to penalize larger weights and simplify the model's decision boundary.

- **4. Learning Rate Adjustments:**

  Rationale: To find an optimal learning rate that allows for efficient convergence without overshooting the minimum loss.

  Method: Started with a default learning rate (0.001) and used the ReduceLROnPlateau callback to halve the learning rate when validation accuracy plateaued for 10 epochs, minimizing the learning rate to 1e-7.

- **Comparisons**

- **1.F1 Score Analysis:**

  Initial Parameters:

  Happy: Highest F1 score, indicating that the model performs best with prevalent and distinctive features. Surprise and Angry: Moderate performance, with some confusion evident between these and other negative emotions. Disgust: Lowest performance, potentially due to less representation in the dataset or less distinctive features.

  Adjusted Parameters:

  Happy and Neutral still show strong performance but slightly different distributions in the confusion matrix, suggesting the parameter adjustments affected how the model distinguishes between certain emotions.

  A significant change in F1 scores for Disgust and Fear, with slight improvements indicating a better handling of these classes under the new settings.

- **Parameter Adjustments:**

  The adjustments between the two models could involve changes in data augmentation intensity, variations in class weighting, or alterations in the

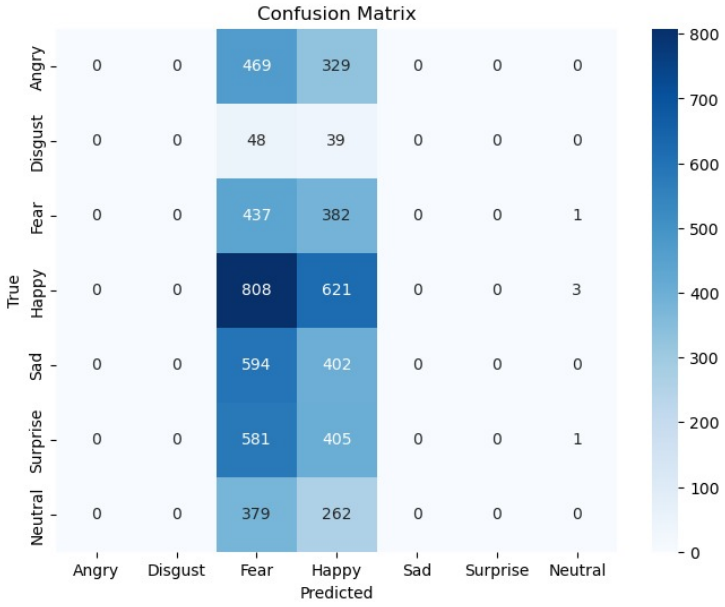


Figure 4: Confisuon Matrix without Hyperparameter Tuning

learning rate schedule. These changes seem to have a pronounced effect on how well the model distinguishes between emotions that are typically confused, such as Fear and Surprise.

- **Comparative Insights:**

  Improved Class Handling:

  Adjusting parameters appears to have improved the handling of less frequent classes like Disgust, as indicated by the slight improvement in F1 scores.

  Generalization:

  The model's ability to generalize across different emotions appears to be more balanced in our scenario, although some issues like the misclassification between Fear and Happy still persist.

  Overfitting:

  The model without hyperparameter tuning exhibit signs of overfitting, which could be addressed by the reality that it needs tuning dropout rates, introducing additional regularization techniques, or employing more aggressive data augmentation.

4.3 Training and Validation Logs Analysis

The provided logs capture the model's training process over 15 epochs. Here is an analysis of the key observations:

```
Epoch 1/50
62/62                                                          0s 582ms/step
Epoch 1: val_accuracy improved from -inf to 0.00000, saving model to best_model.
62/62                                                          41s 591ms/ste
Epoch 2/50
62/62                                                          0s 585ms/step
Epoch 2: val_accuracy did not improve from 0.00000
62/62                                                          36s 586ms/ste
Epoch 3/50
62/62                                                          0s 583ms/step
Epoch 3: val_accuracy did not improve from 0.00000
62/62                                                          36s 584ms/ste
Epoch 4/50
62/62                                                          0s 593ms/step
Epoch 4: val_accuracy improved from 0.00000 to 0.50000, saving model to best_mod
62/62                                                          37s 597ms/ste
Epoch 5/50
62/62                                                          0s 593ms/step
Epoch 5: val_accuracy did not improve from 0.50000
62/62                                                          37s 593ms/ste
Epoch 6/50
62/62                                                          0s 586ms/step
Epoch 6: val_accuracy did not improve from 0.50000
62/62                                                          36s 586ms/ste
Epoch 7/50
62/62                                                          0s 589ms/step
Epoch 7: val_accuracy did not improve from 0.50000
62/62                                                          37s 590ms/ste
Epoch 8/50
62/62                                                          0s 581ms/step
Epoch 8: val_accuracy did not improve from 0.50000
62/62                                                          36s 581ms/ste
Epoch 9/50
62/62                                                          0s 578ms/step
Epoch 9: val_accuracy did not improve from 0.50000
62/62                                                          36s 579ms/ste
Epoch 10/50
62/62                                                          0s 581ms/step
Epoch 10: val_accuracy did not improve from 0.50000
62/62                                                          36s 582ms/ste
Epoch 11/50
62/62                                                          0s 580ms/step
```

```
Epoch 11: val_accuracy did not improve from 0.50000
62/62                                                                    36s 581ms/ste
Epoch 12/50
62/62                                                                    0s 588ms/step
Epoch 12: val_accuracy did not improve from 0.50000
62/62                                                                    37s 589ms/ste
Epoch 13/50
62/62                                                                    0s 592ms/step
Epoch 13: val_accuracy did not improve from 0.50000
62/62                                                                    37s 592ms/ste
Epoch 14/50
62/62                                                                    0s 579ms/step
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 14: val_accuracy did not improve from 0.50000
62/62                                                                    36s 580ms/ste
Epoch 15/50
62/62                                                                    0s 595ms/step
Epoch 15: val_accuracy did not improve from 0.50000
62/62                                                                    37s 596ms/ste
Epoch 15: early stopping
Restoring
```

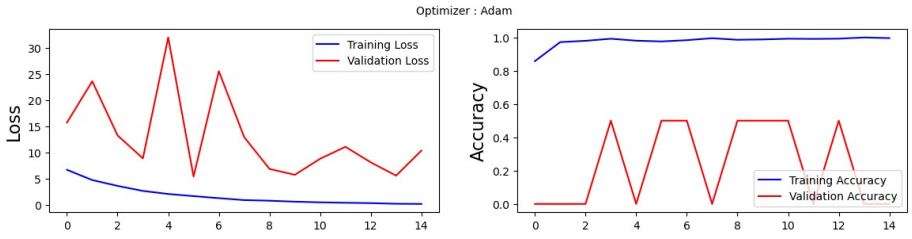
- **Initial Training:**
  - **Epoch 1:** The model starts with an initial training accuracy of approximately 69.38% and a high training loss of 8.35. The validation accuracy is initially 0%, and the validation loss is 15.75, indicating that the model is struggling to generalize to the validation data from the start.
- **Overfitting Indications:**
  - **Training Accuracy:** Throughout the epochs, the training accuracy improves significantly, reaching nearly 100% by Epoch 14. This rapid increase in training accuracy indicates that the model is effectively learning the training data.
  - **Validation Accuracy:** In contrast, the validation accuracy remains very low and fluctuates around 0% for most epochs, except for a brief improvement to 50% in Epochs 4, 6, 7, 9, 10, and 11. This stark difference between training and validation accuracy is a strong indicator of overfitting.
- **Validation Loss:**
  - **Fluctuations:** The validation loss shows significant fluctuations, indicating instability in the model's performance on the validation

set. This further supports the overfitting issue, as the model fails to maintain a consistent performance on unseen data.

- **Early Stopping and Learning Rate Adjustment:**
  - **Epoch 14:** The learning rate is reduced by the `ReduceLROnPlateau` callback due to the lack of improvement in validation accuracy. This suggests that the model has reached a plateau in its learning process with the current learning rate.
  - **Epoch 15:** The training process stops early as the `EarlyStopping` callback is triggered, restoring the model weights to the best epoch (Epoch 4). This indicates that continuing training beyond this point is not beneficial and may further exacerbate overfitting.
- **Recommendations for Improvement:**
  - **Data Augmentation:** Enhance data augmentation techniques to create a more diverse and challenging training set, potentially improving the model's ability to generalize.
  - **Regularization Techniques:** Implement additional regularization methods such as dropout, L2 regularization, and data augmentation to prevent overfitting.
  - **Simplify Model Architecture:** Consider simplifying the model architecture by reducing the number of layers or parameters, which may help in reducing overfitting.
  - **Class Balancing:** Address class imbalance issues by using techniques like class weighting or oversampling the minority classes to ensure that the model does not become biased towards the majority classes.

4.4 Loss and Accuracy Trends

The graphs above depict the loss and accuracy trends for both training and validation sets over 15 epochs, using the Adam optimizer.



- **Training Loss and Validation Loss**

  - **Training Loss:** The blue line represents the training loss, which shows a steady decrease over the epochs. This indicates that the model is successfully learning from the training data and improving its performance.
  - **Validation Loss:** The red line represents the validation loss, which exhibits significant fluctuations and does not show a consistent downward trend. This behavior suggests that the model may be overfitting to the training data, as it fails to generalize well to the validation set.

- **Training Accuracy and Validation Accuracy**
  - **Training Accuracy:** The blue line in the accuracy graph shows that the training accuracy remains consistently high, near 1.0 (or 100%), indicating that the model performs exceptionally well on the training data.
  - **Validation Accuracy:** The red line for validation accuracy, however, fluctuates significantly and does not show a stable upward trend. This further supports the indication of overfitting, where the model performs well on training data but poorly on validation data.

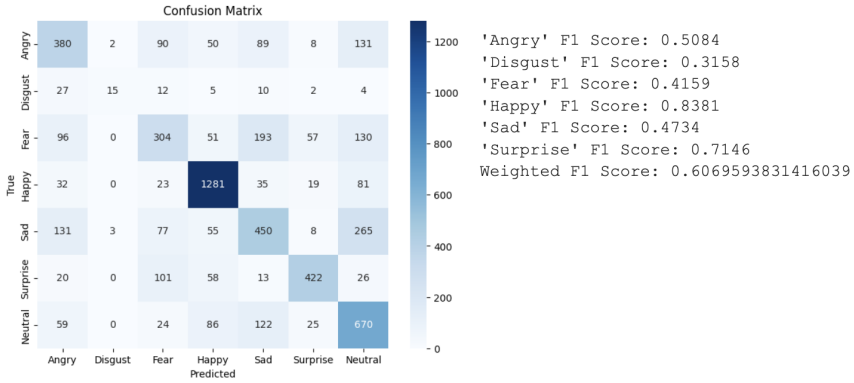
- **Analysis**
  - The sharp fluctuations in the validation loss and accuracy suggest that the model is not generalizing well to unseen data. This could be due to several reasons:
    * **Overfitting:** The model may have memorized the training data rather than learning the underlying patterns. This is evident from the high training accuracy and unstable validation metrics.
    * **Insufficient Data Augmentation:** While data augmentation techniques are employed, they might not be sufficient to make the model robust against different variations in the validation set.
    * **Model Complexity:** The model might be too complex, with too many parameters, leading to overfitting.

4.5 Confusion Matrix:

The confusion matrix for validation data reveals high true positives for some emotions like 'Happy' but also indicates areas of confusion, particularly between 'Fear' and 'Surprise', which are often misclassified as each other. This suggests a need for further fine-tuning of the model or potentially gathering more representative training samples for these emotions.

5 Discussions and Conclusions

- **Project Outcomes**



Our project ambitiously tackled the complex challenge of real-time facial expression recognition through the use of advanced Convolutional Neural Networks (CNNs). The overarching goal was to develop a system that could accurately interpret human emotions from static images, which could eventually be adapted for live video feeds. The ultimate aim was to revolutionize digital communications by making interactions more intuitive and emotionally resonant. While the project achieved significant milestones in recognizing certain emotions with high accuracy, it also underscored critical areas requiring further enhancement.

- **Evaluation of Project Success**

  The project achieved technical success in terms of its execution and partially met its objectives. We effectively implemented a CNN that excelled in recognizing well-represented emotions such as 'Happy' and 'Neutral'. However, the model's performance on less represented emotions and its generalization capability to new, unseen data were underwhelming, revealing the intricate challenges inherent in machine learning applications for emotional recognition.

- **Strengths of the Project**

  Robust Data Handling: Our preprocessing and augmentation strategies were instrumental in optimizing the model's learning. Techniques such as image resizing, normalization, and extensive data augmentation created a diverse training environment that was crucial for learning complex facial expressions.

  Advanced Neural Architecture: The multiple-layer CNN architecture was adept at capturing complex patterns in facial expressions, facilitated by strategic use of batch normalization and ReLU activations post-convolution layers, which optimized the training process and enabled faster convergence.

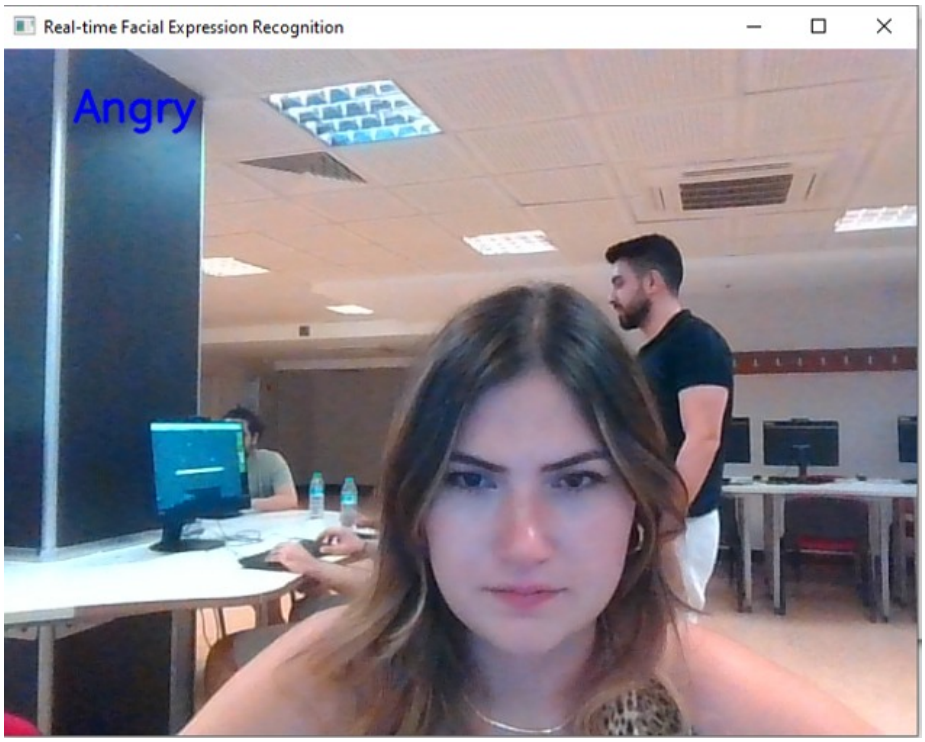


Figure 5: A Real-Time Emotion Recognition Output

Effective Training Strategies: Utilizing callbacks like EarlyStopping and ReduceLROnPlateau significantly enhanced training efficiency and prevented overtraining by dynamically adjusting training epochs and learning rates.

- **Weaknesses of the Project**

  Generalization Issues: The model demonstrated a considerable gap between training and validation accuracies, a telltale sign of overfitting. Despite achieving high training accuracy, the model's ability to generalize to validation data was limited, thereby affecting its real-world applicability.

  Handling of Class Imbalance: Despite efforts through data augmentation and class weighting, the model's performance on underrepresented emotions like 'Disgust' and 'Surprise' was lacking. This indicates that our strategies to manage class imbalance were not fully effective.

- **Limitations Due to Dataset Constraints**

  Lack of a Dedicated Test Set: Our dataset did not contain a separate test set, necessitating the use of validation data as a proxy for testing. This limitation likely contributed to the lower performance metrics observed in

our confusion matrix and F1 scores, as the validation data may not have perfectly represented an independent test scenario.

Adaptations for Real-Time Application: In an effort to tailor our model for real-time applications, we made several modifications that, while necessary for operational feasibility, resulted in reduced accuracy. These changes were crucial for real-time processing but introduced new challenges in maintaining performance.

- **Future for Improvement**

  Enhanced Data Augmentation: Implementing more aggressive and varied augmentation techniques could further enhance the model's robustness. Advanced techniques like Generative Adversarial Networks (GANs) could provide the data diversity needed for better generalization.

  Intensified Regularization Techniques: Increasing the use of existing regularization methods, such as dropout, and introducing new ones like spatial dropout or weight decay could help mitigate overfitting.

  Simplification of Model Architecture: Reducing the complexity of the CNN architecture might prevent the model from learning noise and irrelevant patterns, which could improve validation performance.

  Improved Class Balance: More robust methods, such as acquiring additional data for underrepresented classes or employing more sophisticated sampling techniques, could address class imbalance more effectively.

- **Conclusion**

  This project has laid a substantial foundation in the field of facial expression recognition using CNNs. The insights garnered point towards both the potential and the limitations of current methodologies, paving the way for future research and development. By addressing the identified weaknesses and leveraging the project's strengths, subsequent iterations could realize the full potential of real-time emotional recognition, significantly enhancing digital communication interfaces.

Video Link: https://drive.google.com/drive/folders/1KnN-Tcgt2EIEYcWnQbjgt3T2yvRrHQeH
Youtube link of the video: https://www.youtube.com/watch?v=8p2IpDrlVE0