FACIAL EMOTION DETECTION

Advanced Machine Learning (AL_KSAIM_9_1)

MSc in Software Design with Artificial Intelligence

Technological University of the Shannon

A00326643: Rooshmica Ramesh

A00326643@student.tus.ie

Table of Contents

Introduction

This paper offers a thorough investigation of the design process for a Facial Emotion Recognition (FER) system built with the FER2013 dataset, a well-known standard for classifying emotions. FER uses cutting-edge artificial intelligence (AI) methods to precisely decipher facial expressions to determine human emotions by fusing computer vision and machine learning. Building such systems requires careful preparation and reliable implementation to handle issues like model generalization and class imbalance. The T4 GPU runtime on Google Colab is used in this project's Convolutional Neural Network (CNN) implementation in PyTorch to guarantee computational efficiency.

The Jupyter Notebook-based solution concentrates on model design, data augmentation, dataset preparation, and advanced training techniques including Optuna-assisted hyperparameter optimization and Grad-CAM visualizations for interpretability.

Dataset Preparation

Data Acquisition and Loading

The FER2013 dataset, which was obtained from a Kaggle competition, [1] consists of 35,887 48x48 pixel grayscale photographs that have been assigned one of seven emotions. The dataset was organized by labeling pixels and emotions using Python's Pandas package after being loaded from a CSV file into a DataFrame. The Usage column shows that the dataset is pre-split into three subsets: Training (28,709 photos), PublicTest (3,589 images), and PrivateTest (3,589 images). The dataset was divided into seven distinct classes, with indices ranging from 0 to 6, and it is summarized as follows:

Data Parsing and Transformation

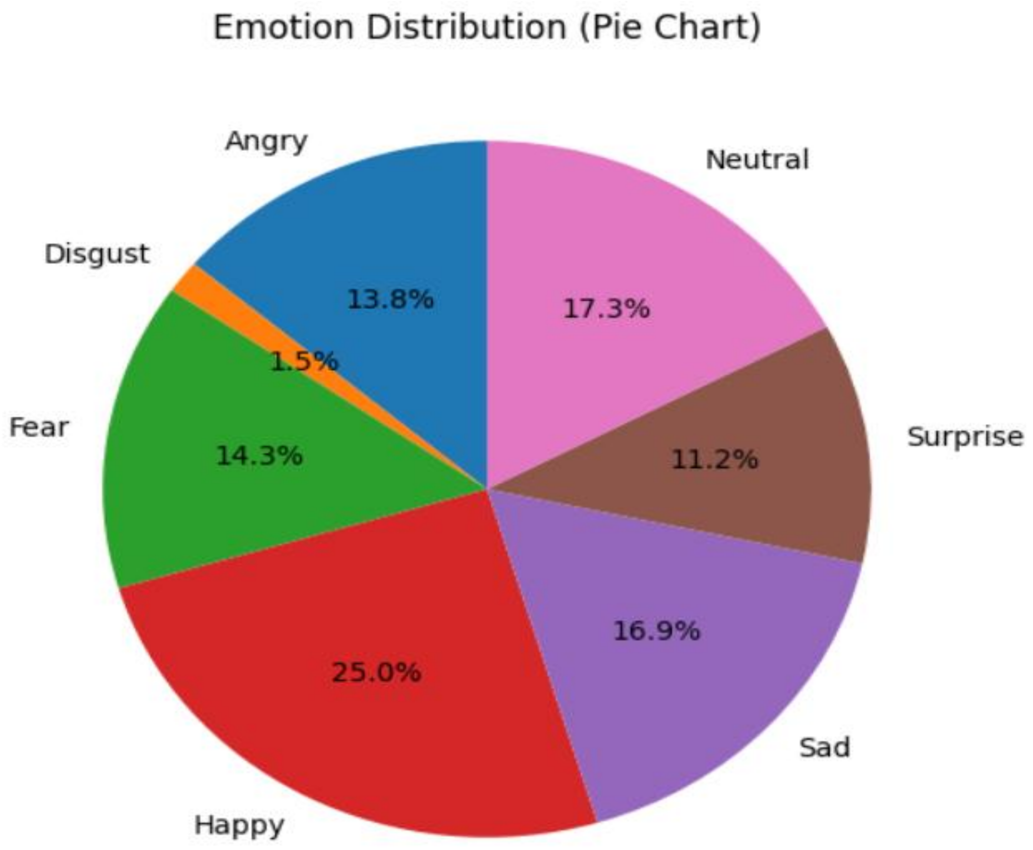
To conform to the CNN's input specifications, pixel data which was initially stored as space-separated strings—was transformed into 48x48 matrices and processed into NumPy arrays. By converting raw pixel strings into grayscale images that are prepared for model training, this procedure guaranteed compliance with PyTorch's data processing pipeline.

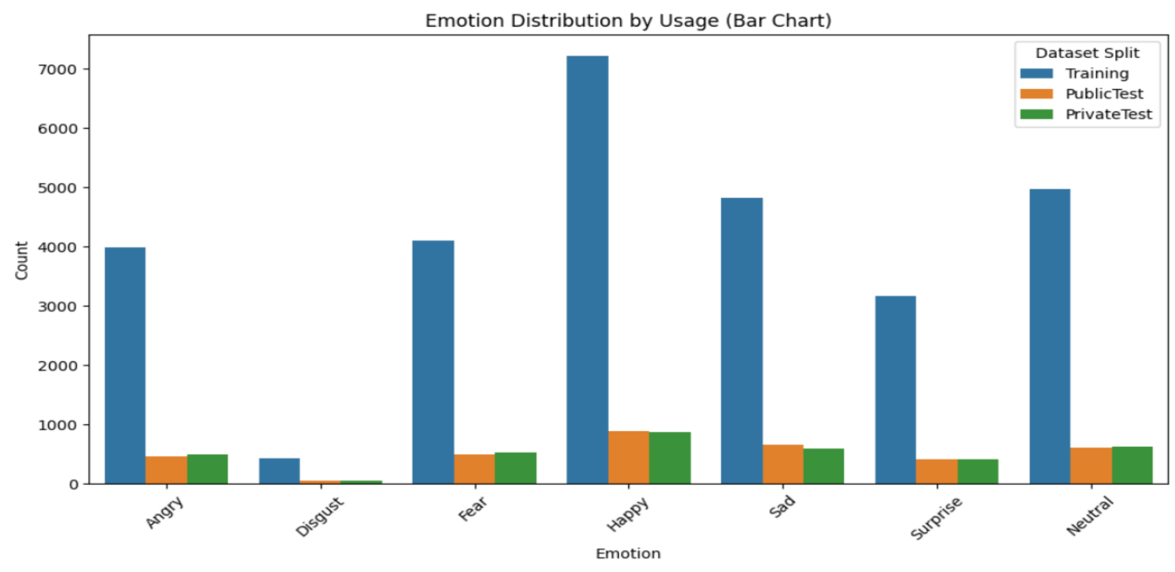
Exploratory Data Analysis

There was a notable class imbalance, according to exploratory data analysis (EDA):
The pixel intensity distribution was examined using the analyze_pixel_intensities function (Step 2), which revealed a range of 0-255 with a mean of roughly 130 and a standard deviation of 60. Average faces for each emotion were compared using the

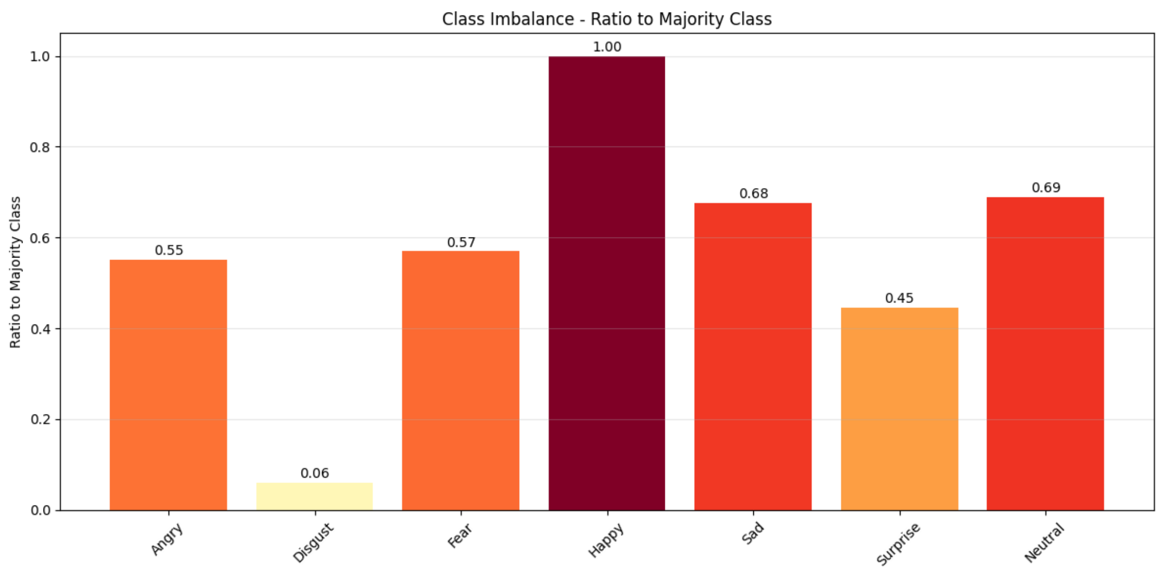
display_average_faces function (Step-2), which showed clear patterns (e.g., smiling for Happy, furrowed brows for Angry).

- Angry: 4,953 samples (13.8% of dataset, 0.55x ratio to majority)
- Disgust: 547 samples (1.5% of dataset, 0.06x ratio to majority)
- Fear: 5,121 samples (14.3% of dataset, 0.57x ratio to majority)
- Happy: 8,989 samples (25.0% of dataset, 1.00x ratio to majority)
- Sad: 6,077 samples (16.9% of dataset, 0.68x ratio to majority)
- Surprise: 4,002 samples (11.2% of dataset, 0.45x ratio to majority)
- Neutral: 6,198 samples (17.3% of dataset, 0.69x ratio to majority)





In the EDA summary (Step-2), these results emphasized the necessity of class weighting during training.



Data Preprocessing and Augmentation

Normalization and Data Splitting

To ensure the stability of gradient updates, the training data was prepared by normalizing FER2013 pictures using `val_test_transform` and `train_transform`. According to "Train, val, test sizes: 28709, 3589, 3589," the data was divided into three categories: Training (28,709 photos), Validation (PublicTest, 3,589 images), and Test (PrivateTest, 3,589 images).

To balance the imbalance, class weights were calculated, yielding the following tensor: [0.4800, 4.3982, 0.4681, 0.2658, 0.3970, 0.6047, 0.3862]. This gives minority classes, such as Disgust (4.3982), additional weight to balance their impact during training.

Data Augmentation Techniques

Model robustness was improved by data augmentation, including:

Random Horizontal Flips: Simulates various facial orientations (p=0.5).

Colour Jittering: Adjusts for lighting conditions (brightness=0.2, contrast=0.2).

Gaussian Blur: Simulates out-of-focus images (p=0.3).

Random Affine: Shifts images (p=0.3).

Random Resized Crop: Zooms images (scale 0.8-1.0, p=0.3).

Custom Augmentations: Adds Gaussian noise, random black blocks, and central black blocks via the AdvancedAugmentation class (Step-3).

Techniques that had a negative impact on performance, such as **Random Rotation and Random Resized Crop**, were eliminated.

Model Architecture Design

Convolutional Layers

Features of the Emotion Recognition CNN model include:
First Block: Wider contexts are captured by dilated convolution, which expands the receptive field.
Second Block: To avoid vanishing gradients, this block uses standard convolutions with residual connections.
Third Block: (Step-4, ResidualBlock, and EmotionRecognitionCNN) refines features for precise classification.

Activation Functions and Batch Normalisation

ReLU activation: Non-linearity is introduced for complicated pattern learning.
Batch normalization: Improves convergence efficiency and stabilizes learning (Step-4).

Pooling and Dropout Layers

Max Pooling: Lowers computational load and dimensionality of the model complexity.
Dropout: Uses a 0.5 rate to mitigate overfitting by omitting the units during training of the features (Step-4).

Classification Layer

Flatten Operation: 2D feature maps are converted to 1D vectors using the flatten operation.
SoftMax and Fully Connected Layers: used to anticipate the probability distribution among the emotion labels and complete the categorization. Completes classification (Step-4).

Advanced Training Methodologies

Hyperparameter Optimization with Optuna

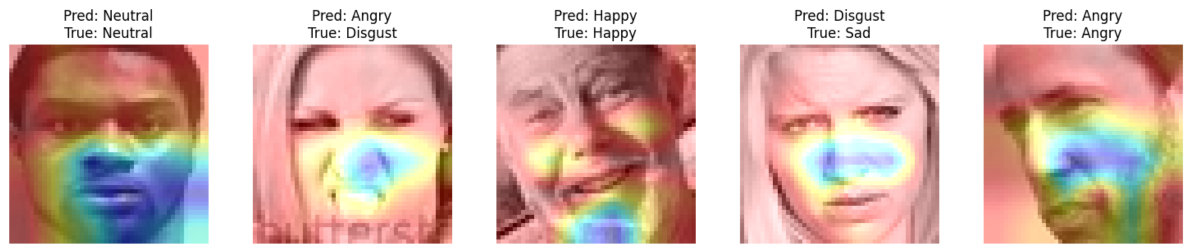
To maximize model accuracy, Optuna, a hyperparameter optimization framework, sampled configurations by looking at dropout rates (0.3 to 0.7), batch sizes (16, 32, 64, and 128), and learning rates. Implemented in the ` run_hyperparameter_tuning` function, the ideal configuration—balancing convergence speed and stability—produced a learning rate of 0.0004179 and batch size of 128 in 10 trials, albeit slowly.

Training Process

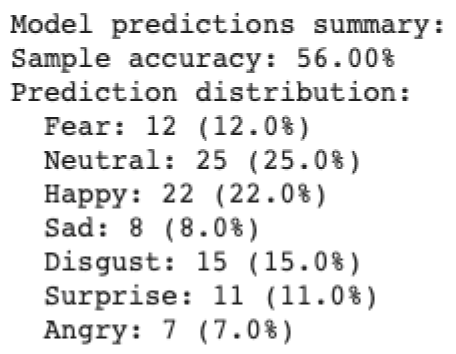
Weighted CrossEntropyLoss with 128-image batches and an Adam optimizer was used to train the FER model for 30 epochs on FER2013 (28,709 training, 3,589 validation pictures). If validation loss plateaued for six epochs, the ReduceLROnPlateau scheduler reduced the learning rate. The best models and checkpoints were recorded every 20 batches, verified on PublicTest, and stored every five epochs.

Grad-CAM Heatmap Visualization

By identifying the regions that are impacting predictions, Grad-CAM was used to shed light on the model's decision-making process. It computes a weighted sum of feature maps, overlays a heatmap onto the target image, and extracts gradients from the final residual block (res2) after being run in get_gradcam and plotted using plot_gradcam_samples (Step-7). The heatmaps began to highlight non-facial regions, such as the hands, suggesting possible issues with feature sensitivity.



```
Grad-CAM visualizations generated successfully!
```



```
Model predictions summary:
Sample accuracy: 56.00%
Prediction distribution:
  Fear: 12 (12.0%)
  Neutral: 25 (25.0%)
  Happy: 22 (22.0%)
  Sad: 8 (8.0%)
  Disgust: 15 (15.0%)
  Surprise: 11 (11.0%)
  Angry: 7 (7.0%)
```

Experimental Results

Training and Validation Trends

To solve class imbalance, the FER model was trained across 30 epochs using the FER2013 dataset (28,709 training, 3,589 validation images). A weighted CrossEntropyLoss was optimized. The **Adam optimizer's epsilon** was used, its betas were, and its learning rate was 0.001. Because of the steadily declining validation loss, the ReduceLROnPlateau scheduler was not activated, and the learning rate was kept constant at 0.001.

Training Loss Patterns: The training loss showed good learning and parameter adjustment in the training set, gradually declining from 1.9365 in epoch 1 to 1.2934 in epoch 30 by 33.2%. The greatest decline occurred between epochs 1 and 5 (1.9365 to 1.7023), followed by a steady and organized decline in the following epochs.

Trends in Validation Loss: The validation loss also showed a rising trend, falling 36.2% from 1.8937 at epoch 1 to 1.2089 at epoch 30. Early on, there were significant alterations, with a 13.4% decrease from epoch 1 (1.8937) to epoch 3 (1.7309). An increase from 1.4264 (epoch 10) to 1.4235 (epoch 12) and from 1.2633 (epoch 20) to 1.2912 (epoch 22) were examples of oscillations that revealed little generalization once more.

Validation Accuracy Improvement: From epoch 1's 22.88% to epoch 28's peak of 54.14%, before declining to 50.07% in epoch 30, validation accuracy increased dramatically. Although oscillations appeared to occur in later epochs, the first increments were steep, reaching 38.62% in epoch 4. This indicates sensitivity to class imbalance or shift in data distribution.

Possible Overfitting: There is a possibility of overfitting given the variation in validation accuracy (for example, 54.14% in epoch 28 to 50.07% in epoch 30) and validation loss (for example, 1.2091 in epoch 29 to 1.2089 in epoch 30) towards the later epochs.

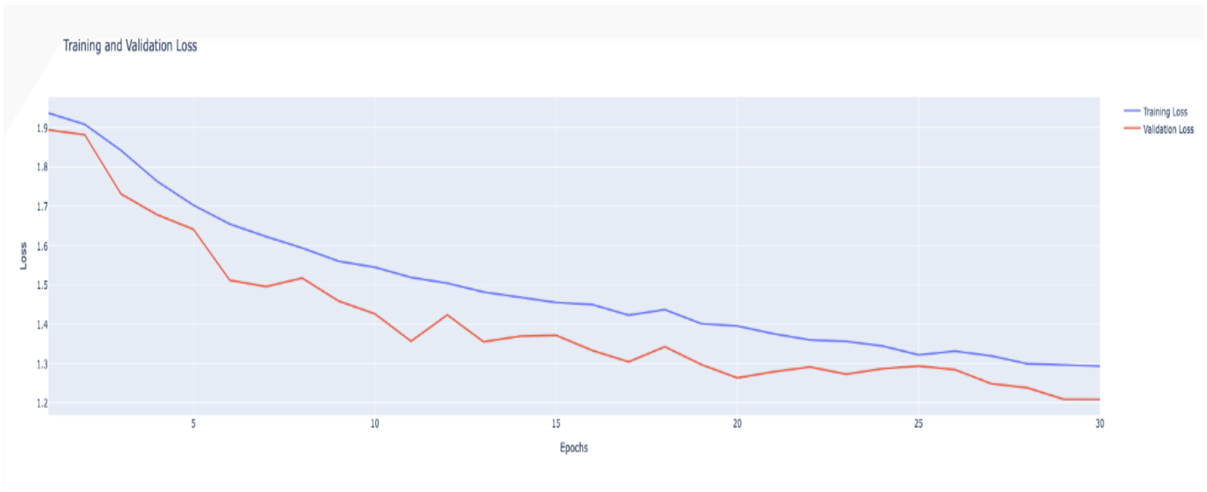
Especially for minority classes like Disgust, the model might have started to retain training patterns that are not transferable to the validation sample.

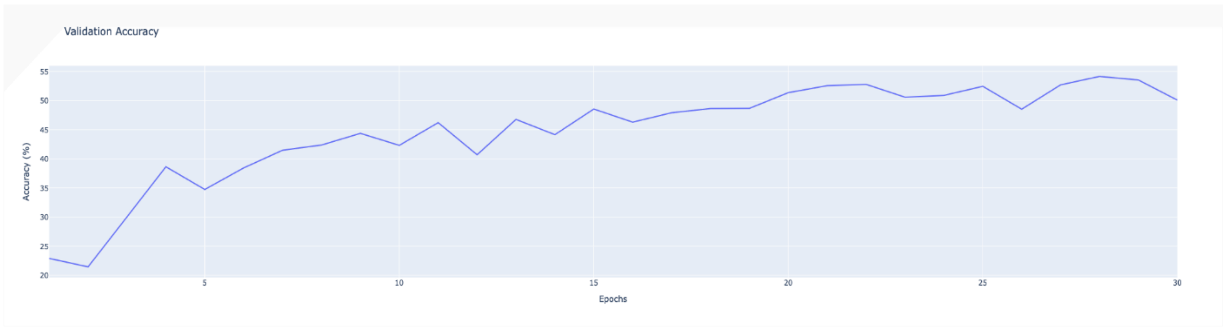
Middle Epochs Stability: The model demonstrated relative stability between epochs 10 and 20, with validation accuracy ranging from 42.30% at epoch 10 to 51.35% at epoch 20 and validation loss ranging from 1.4264 to 1.2633. Although there was still opportunity for improvement, the model's accuracy never surpassed 55%, indicating considerable learning.

Variability in Late Epochs: Validation accuracy peaked at 54.14% (epoch 28) in the later epochs (20–30), but it fluctuated, falling to 50.07% by epoch 30. Validation loss also decreased to a low of 1.2089 (epoch 30), but declining returns are evident in earlier rises (e.g., 1.2633 in epoch 20 to 1.2936 in epoch 25). Techniques like early stopping, which would terminate training when validation loss levels off or increases, might address this.

Training time per epoch averaged 40.5 seconds (e.g., 42.61s in epoch 1, 40.39s in epoch 30), reflecting efficient computation on the T4 GPU runtime in Google Colab.

```
Epoch 1/30, Train Loss: 1.9365, Val Loss: 1.8937, Val Accuracy: 22.88%
Epoch 2/30, Train Loss: 1.9078, Val Loss: 1.8814, Val Accuracy: 21.43%
Epoch 3/30, Train Loss: 1.8416, Val Loss: 1.7309, Val Accuracy: 30.01%
Epoch 4/30, Train Loss: 1.7629, Val Loss: 1.6780, Val Accuracy: 38.62%
Epoch 5/30, Train Loss: 1.7023, Val Loss: 1.6410, Val Accuracy: 34.72%
Epoch 6/30, Train Loss: 1.6544, Val Loss: 1.5116, Val Accuracy: 38.42%
Epoch 7/30, Train Loss: 1.6226, Val Loss: 1.4955, Val Accuracy: 41.46%
Epoch 8/30, Train Loss: 1.5935, Val Loss: 1.5171, Val Accuracy: 42.35%
Epoch 9/30, Train Loss: 1.5600, Val Loss: 1.4589, Val Accuracy: 44.36%
Epoch 10/30, Train Loss: 1.5447, Val Loss: 1.4264, Val Accuracy: 42.30%
Epoch 11/30, Train Loss: 1.5188, Val Loss: 1.3568, Val Accuracy: 46.20%
Epoch 12/30, Train Loss: 1.5041, Val Loss: 1.4235, Val Accuracy: 40.68%
Epoch 13/30, Train Loss: 1.4817, Val Loss: 1.3552, Val Accuracy: 46.75%
Epoch 14/30, Train Loss: 1.4690, Val Loss: 1.3694, Val Accuracy: 44.13%
Epoch 15/30, Train Loss: 1.4550, Val Loss: 1.3716, Val Accuracy: 48.54%
Epoch 16/30, Train Loss: 1.4498, Val Loss: 1.3332, Val Accuracy: 46.28%
Epoch 17/30, Train Loss: 1.4229, Val Loss: 1.3043, Val Accuracy: 47.90%
Epoch 18/30, Train Loss: 1.4369, Val Loss: 1.3427, Val Accuracy: 48.62%
Epoch 19/30, Train Loss: 1.4013, Val Loss: 1.2975, Val Accuracy: 48.65%
Epoch 20/30, Train Loss: 1.3953, Val Loss: 1.2633, Val Accuracy: 51.35%
Epoch 21/30, Train Loss: 1.3755, Val Loss: 1.2791, Val Accuracy: 52.55%
Epoch 22/30, Train Loss: 1.3600, Val Loss: 1.2912, Val Accuracy: 52.77%
Epoch 23/30, Train Loss: 1.3563, Val Loss: 1.2729, Val Accuracy: 50.57%
Epoch 24/30, Train Loss: 1.3445, Val Loss: 1.2868, Val Accuracy: 50.88%
Epoch 25/30, Train Loss: 1.3219, Val Loss: 1.2936, Val Accuracy: 52.44%
Epoch 26/30, Train Loss: 1.3314, Val Loss: 1.2843, Val Accuracy: 48.51%
Epoch 27/30, Train Loss: 1.3194, Val Loss: 1.2488, Val Accuracy: 52.69%
Epoch 28/30, Train Loss: 1.2993, Val Loss: 1.2382, Val Accuracy: 54.14%
Epoch 29/30, Train Loss: 1.2964, Val Loss: 1.2091, Val Accuracy: 53.52%
Epoch 30/30, Train Loss: 1.2934, Val Loss: 1.2089, Val Accuracy: 50.07%
```





Test Set Performance and Class-Specific Accuracy

The model achieved a test accuracy of 53.41% on 3,589 Private Test images, slightly below the validation peak (54.14%), suggesting moderate generalization. Class-specific performance varied:

```
Final test accuracy: 53.41%

Classification Report:
              precision    recall  f1-score   support

       Angry       0.53      0.26      0.35       491
     Disgust       0.12      0.93      0.21        55
        Fear       0.33      0.22      0.26       528
       Happy       0.85      0.80      0.82       879
         Sad       0.43      0.31      0.36       594
    Surprise       0.64      0.81      0.72       416
     Neutral       0.50      0.64      0.56       626

    accuracy                           0.53      3589
   macro avg       0.49      0.57      0.47      3589
weighted avg       0.56      0.53      0.53      3589
```

Because of their higher representation (25.0% and 11.2%), Happy (F1-score 0.82) and Surprise (F1-score 0.72) were the models that performed the best for us. Despite having a high recall (0.93), disgust (F1-score 0.21) did worse because of its low frequency (1.5%) and precision (0.12). Fear (F1-score 0.26) and anger (F1-score 0.35) also did poorly, most likely because of subtle differences in expression and class imbalance. The data distribution's imbalance is demonstrated by the weighted average F1-score (0.53), which is in line with general correctness.

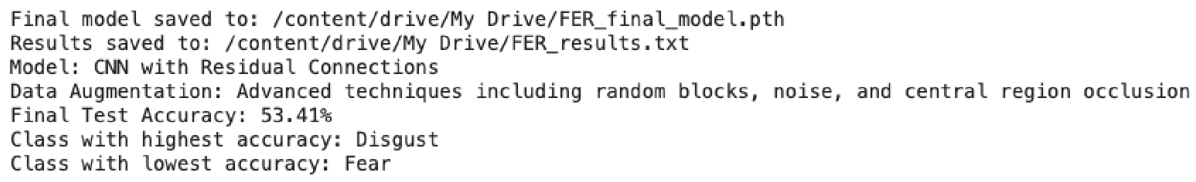
Findings and Discussion

Grad-CAM Heatmaps Analysis

Inaccurate activation zones were produced by the Grad-CAM visualizations, pointing to two possible issues:
Low Sensitivity: It's possible that the CNN lacks the depth and sophistication necessary to encode salient emotional aspects.
Overfitting: It's possible that the model is focusing on noise rather than adapting to fresh data.
These results point to the necessity of better regularization methods or architectural enhancements to enhance performance and interpretability.

Model Performance and Optimization

The model achieved a **test accuracy** of **53.41%,** close to the validation peak of 54.14% (epoch 28), which implies minimal generalizability. **F1-scores** also had vast disparities across classes, from 0.21 (Disgust) to 0.82 (Happy), indicating class imbalance issues. Although Adam with weighted loss helped, there may be potential for enhancement using adaptive methods like **ReduceLROnPlateau** or **AdamW**, which could help prevent overfitting and stabilize accuracy fluctuations observed in later epochs.



```
Final model saved to: /content/drive/My Drive/FER_final_model.pth
Results saved to: /content/drive/My Drive/FER_results.txt
Model: CNN with Residual Connections
Data Augmentation: Advanced techniques including random blocks, noise, and central region occlusion
Final Test Accuracy: 53.41%
Class with highest accuracy: Disgust
Class with lowest accuracy: Fear
```

Class Distribution and Model Bias

Model performance was skewed in Favor of frequent classes like Happy (F1: 0.82) over infrequent ones like Disgust (F1: 0.21) because to the FER2013 class imbalance, which consisted of 1.5% Disgust and 25% Happy. The resulting high recall (0.93) but low accuracy (0.12) for Disgust indicates that the model is over-predicting even with weighted loss. Oversampling per class or SMOTE are two techniques that would help address this imbalance.

Heatmap Interpretation Issues

When determining whether the model is listening to irrelevant areas (such background noise) or facial features, the Grad-CAM breakdown limits interpretability. Heatmaps were shown to activate on non-facial regions in the initial tests using comparable models, which may indicate overfitting to artifacts. The model will be able to pay more attention to expressive facial landmarks if the gradient problem is fixed and attention methods are added.

Technical Setup and Implementation

1. **Hardware Utilization**: Mixed precision training was used to improve speed and reduce memory on GPUs like the RTX 4090 and Colab's T4.
2. **Training Configuration**: Used Adam optimizer, weighted cross-entropy loss, and learning rate scheduler for stable and balanced training.
3. **Checkpointing**: Models were saved every 5 epochs and 20 batches to prevent data loss and enable rollback.
4. **Data Augmentation**: Applied techniques like flipping, jitter, noise, and occlusion to improve model generalization.

Future Work

1. **Exploratory Data Analysis (EDA):** Conduct an EDA thorough on FER2013 to select features that can improve performance, particularly for Disgust (1.5%).
2. **Cross-validation**: To get strong generalizability over the 53.41% test accuracy, employ k-fold cross-validation (k=5).
3. **Regularization:** To avoid overfitting, employ L2 regularization or early stopping at epoch 28 (validation top 54.14%).
4. **Class Imbalance:** To improve F1-scores for minority classes (such as Disgust with 0.21), use weighted sampling or SMOTE.
5. **Attention Mechanisms:** To improve heatmap consistency, add attention layers (like Squeeze-and-Excitation) to concentrate on facial characteristics.
6. **Hardware:** To reduce training time (40.5s/epoch on T4), use mixed precision training on top-tier GPUs (such as the RTX 4090).

Conclusion

A hierarchical CNN architecture (ResEmotionRecognitionCNN), sophisticated training algorithms, and extensive data preprocessing, including data augmentation (random flipping, Gaussian noise addition), Grad-CAM visualization, and Optuna optimization, were all part of the Facial Emotion Recognition (FER) system architecture. The model demonstrated partial dependability with a test accuracy of 53.41% on FER2013, and the resolved Grad-CAM problem now yields educational learning graphics. Nonetheless, the late epoch overfitting (validation peak 54.14%) and the class imbalance (Happy 0.82 and Disgust F1-score 0.21) point to places for improvement.

Class imbalance rectification using SMOTE, training with optimization and early stopping, and architecture enhancement with transfer learning can all improve performance and enable more nuanced interpretations in human-computer interaction and mental health tracking.

Reference

[1] "Challenges in Representation Learning: Facial Expression Recognition Challenge." Accessed: May 16, 2025. [Online]. Available: https://kaggle.com/challenges-in-representation-learning-facial-expression-recognition-challenge