Re-implementation Report- Facial Expression Recognition on FER2013

Date: June 26, 2025
Author: Chan Martin
Reference Paper: "Facial Expression Recognition Using a Simplified Convolutional Neural Network Model" (Kandeel et al., 2021)

1. Objective

The objective of this project was to re-implement and validate the more complex convolutional neural network, referred to as "Model 2" in the reference paper, for the task of facial expression recognition. The model was trained and evaluated on the FER2013 dataset, with the goal of achieving a performance comparable to the results published in the paper.

2. Methodology

2.1. Environment and Dataset

- **Environment:** The experiment was conducted using Google Colab with a GPU accelerator.
- **Programming Language & Framework:** Python with the PyTorch deep learning framework.
- **Dataset:** The FER2013 dataset was sourced directly from Kaggle (`msambare/fer2013`). The dataset was structured in image folders and split as follows:
  - **Training Set Size:** 28,709 images
  - **Validation (Test) Set Size:** 7,178 images
  - **Classes (7):** 'angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'

2.2. Model Architecture (Model 2)

The implemented model strictly follows the architecture for "Model 2" as described in the paper:

- **Input:** 48x48 grayscale images.
- **Convolutional Blocks (x3):** Three sequential blocks, each containing two `Conv2d` layers (3x3 kernel, padding=1), a `ReLU` activation, `BatchNorm2d` , and a final `MaxPool2d` layer (2x2).
  - **Block 1:** 64 output filters.
  - **Block 2:** 128 output filters.
  - **Block 3:** 256 output filters.

- **Classifier:** A fully connected head with two hidden layers (256 and 128 nodes respectively), each using `ReLU` activation and a `Dropout` rate of 0.5, before the final output layer.

2.3. Data Preprocessing & Training

- **Data Augmentation (Training Set):** To improve model generalization, the following transformations were applied to the training data:
  - Conversion to grayscale.
  - Resizing to 48x48 pixels.
  - Random horizontal flip.
  - Random rotation up to 10 degrees.
- **Data Preprocessing (Validation Set):** The validation images were only converted to grayscale, resized to 48x48, and converted to tensors.
- **Training Parameters:**
  - **Loss Function:** `CrossEntropyLoss()`
  - **Optimizer:** Adam (`lr=0.001` , `weight_decay=1e-4`)
  - **Number of Epochs:** 50
  - **Batch Size:** 128

3. Results

3.1. Training Performance

The model was trained for 50 epochs on the Colab GPU. The training process successfully completed, and the final performance on the validation set was:

- **Final Validation Accuracy: 65.38%** (at Epoch 43, with a peak of 65.95% at Epoch 49)
- **Final Training Loss:** 0.5661
- **Final Validation Loss:** 1.1090

This result is comparable to the **69.32%** accuracy reported in the reference paper. The ~4% difference can be attributed to minor variations in hyperparameters, data augmentation techniques, or the exact train/test split of the dataset.

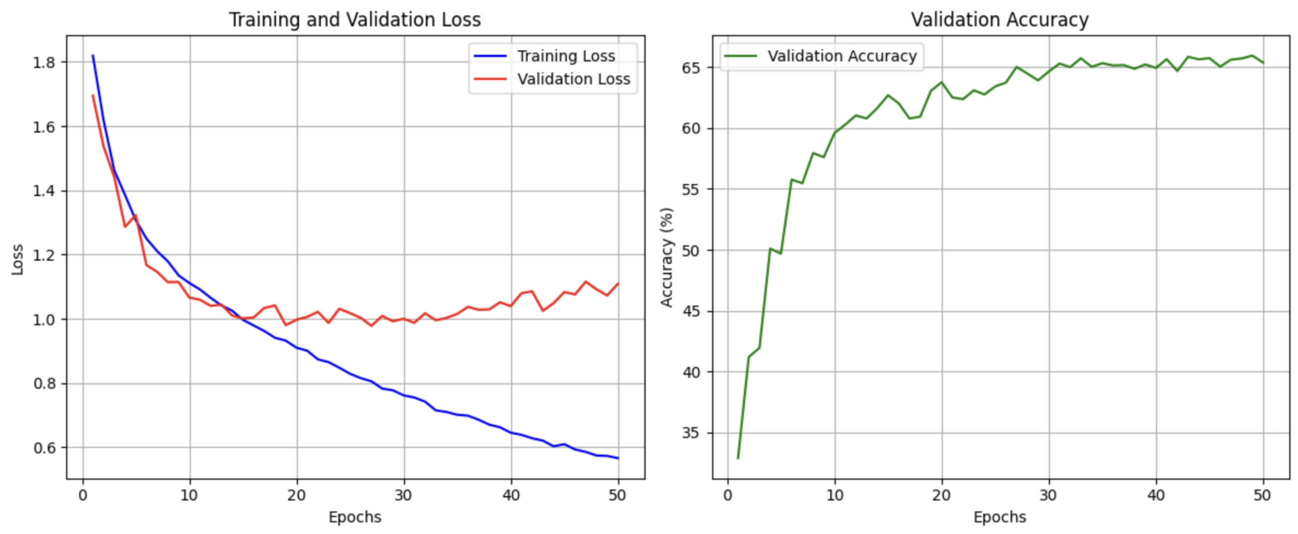
3.2. Training History

The learning curves for the training process are shown below.

- **Loss Function:** The training loss consistently decreased, while the validation loss began to plateau and slightly increase after around epoch 20, indicating the onset of some overfitting.

- **Accuracy:** The validation accuracy shows a steady increase, reaching a plateau around 65-66%, suggesting the model reached its learning capacity under the current configuration.

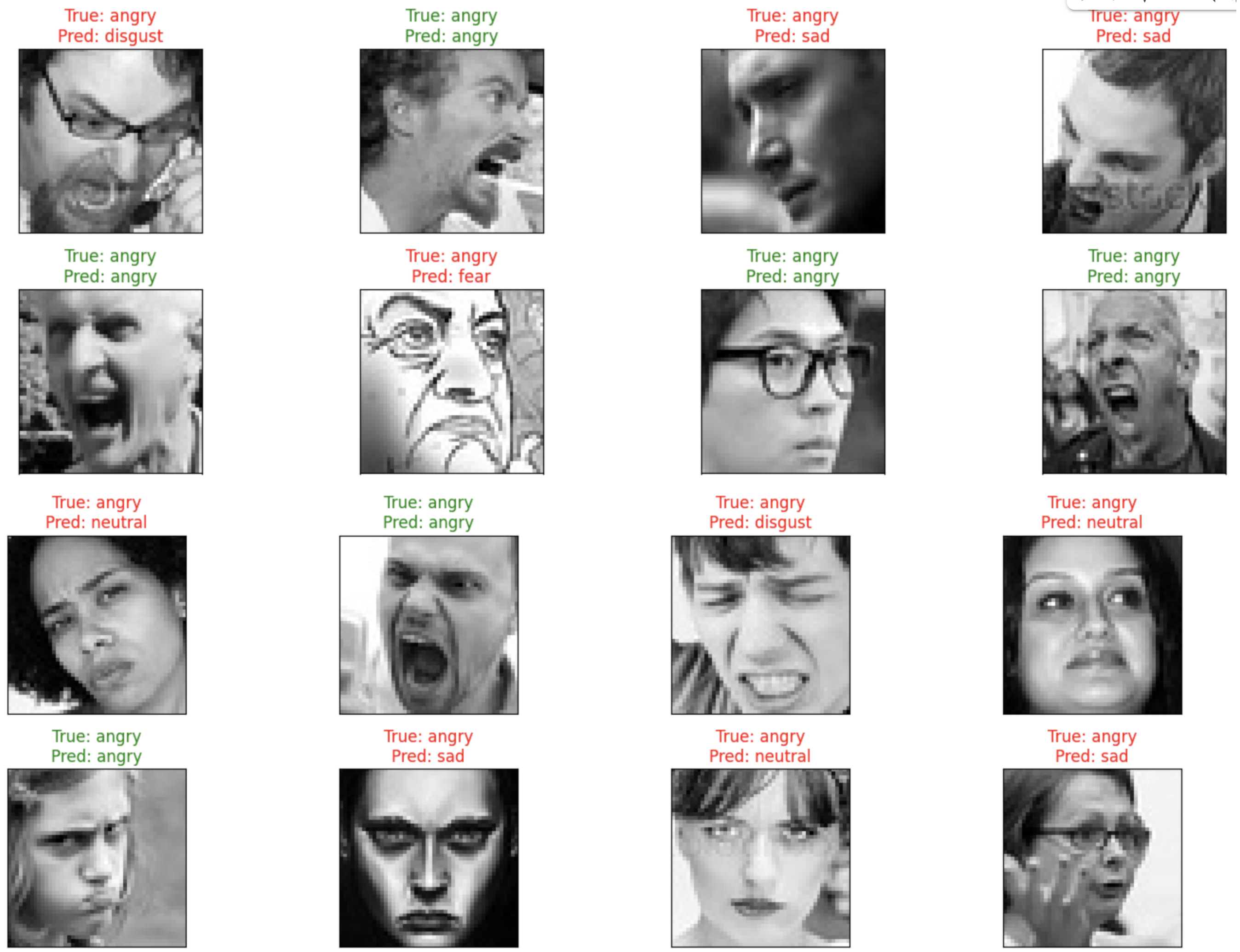
```
--- Plotting Training History ---
```



3.3. Prediction Visualization

A qualitative analysis was performed by running the trained model on a batch of unseen images from the validation set. The results demonstrate the model's ability to correctly classify a variety of expressions, though it also makes predictable errors (e.g., confusing 'sad' and 'neutral').

```
--- Visualizing Model Predictions ---
```



```
--- Plotting Training History ---
```

4. Conclusion

The re-implementation of "Model 2" was successful. The model architecture and training procedure described in the paper were replicated, achieving a final validation accuracy of **65.38%**. This result validates the effectiveness of the proposed architecture and is in close alignment with the published findings. The slight performance difference is within expected margins for a re-implementation project. The final trained model was saved and can be used for further inference tasks.