Emotion Detection from Facial Images

Adam Goldstein, Taran Elgin, Lucie Martin

Abstract

This project aims to develop a Convolutional Neural Network (CNN) for classifying facial expressions into seven emotion categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. Using the FER-2013 dataset, we will preprocess images and experiment with a custom CNN. The model's performance will be evaluated using accuracy, precision, recall, and F1-score. This system has applications in enhancing AI interfaces, assisting psychologists, and monitoring emotional states in high-risk environments.

1 Introduction

Emotion detection from facial images is a growing area of interest with applications in human-computer interaction, psychological assessment, and safety monitoring. By accurately identifying emotions, systems can respond adaptively to user states—improving user experience, detecting distress signals, and analyzing reactions in critical scenarios. In this project, we focus on developing a custom CNN from scratch tailored to the FER-2013 dataset. Our goal is to achieve high classification accuracy while maintaining computational efficiency.

2 Background

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily used for image-processing tasks. They are ideally suited for this task due to their ability to automatically extract hierarchical spatial features (e.g., edges, textures, and facial landmarks). They are composed of multiple layers—each designed to learn increasingly abstract representations of the input. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks while further reducing the parameters required to set up the model.

CNNs leverage convolution operations to extract spatial features from images. Non-linear activation functions (such as the $\text{ReLU} = \max(0, z)$) are applied after convolution to introduce non-linearity. Pooling layers then reduce the spatial dimensions while retaining important features. These operations are repeated over multiple layers, with connected layers at the end producing the final classification outputs.

Backpropagation is used to update the network parameters by minimizing a loss function (often a cross-entropy loss function for classification tasks) using optimization algorithms like stochastic gradient descent (SGD) or Adam. In our report, we focus on these key components and the mathematical theory that justifies their use in image classification tasks.

3 Mathematical Formulation

3.1 Convolutional Operation

The convolutional layers apply a set of trainable filters to extract features from the input image. The convolutional operation for image I with filter K is:

$$S(i,j) = (I * K)(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n),$$

where:

$$\begin{aligned} I(i,j) &: \text{The pixel value at position } (i,j) \text{ in the input image,} \\ K(m,n) &: \text{The weight of the kernel at position } (m,n), \\ S(i,j) &: \text{The output feature map at position } (i,j). \end{aligned}$$

3.2 ReLU Activation Function

ReLU (rectified linear unit) introduces non-linearity into the model:

$$f(x) = \max(0, x).$$

This helps with gradient propagation by ensuring the negative values are 0 while keeping the positive values unchanged.

3.3 Pooling Operation

Max pooling is used to reduce the dimensions of the feature maps while retaining important information by selecting the maximum value within a window.

$$P(x,y) = \max_{(i,j) \in p \times p} F(x+i, y+j),$$

where $F(x,y)$ represents the feature map values.

3.4 Softmax Function

The output layer applies a fully connected (dense) layer, followed by the softmax function to obtain probabilities for the seven emotion categories:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{7} e^{z_j}},$$

where z_i is the output of the last fully connected layer for class i.

3.5 Loss Function

Since we are working with multi-class classification, the loss function is categorical cross-entropy:

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{7} y_{i,j} \log(\hat{y}_{i,j}),$$

where

$$\begin{aligned} N &: \text{ number of samples,} \\ y_{i,j} &: \text{ true label (1 for correct class, 0 otherwise),} \\ \hat{y}_{i,j} &: \text{ predicted probability for class } j. \end{aligned}$$

3.6 Backpropagation and Gradient Descent

The model parameters (weights W, biases b) are updated using the Adam optimizer, which updates parameters based on adaptive learning rates:

$$W^{(t+1)} = W^{(t)} - \eta \left(\frac{m_t}{\sqrt{v_t} + \epsilon} \right),$$

where

$$\begin{aligned} \eta &: \text{ learning rate,} \\ m_t &: \text{ moving average of gradients,} \\ v_t &: \text{ moving average of squared gradients,} \\ \epsilon &: \text{ small value for numerical stability.} \end{aligned} \tag{1}$$

4 Dataset

For this project, we use the FER-2013 dataset, which contains 35,887 grayscale images (48x48 pixels) labeled across seven emotion categories: anger, disgust, fear, happiness, sadness, surprise, and neutral. This dataset, introduced during the ICML 2013 conference, is widely regarded as a benchmark for facial emotion recognition tasks. Ethical concerns taken into account include the potential misuse of emotion detection technology for surveillance and privacy breaches. We will ensure no personal identifiers are included in our dataset.

4.1 Preprocessing Steps

- **Data Extraction and Organization:** Extract data from the archive file and structure files into train and test directories.
- **Resizing:** Ensure all images are 48x48 pixels to ensure uniformity.
- **Normalization:** Scale pixel values between 0 and 1 to facilitate faster model convergence during training.
- **Data Augmentation:** Apply random rotations, flips, width and height shifts, and brightness adjustments to increase dataset variability.
- **Handling Missing Data:** Dataset does not contain missing images based on file count.
- **Data Splitting:** Divide the dataset into training (80%), validation (10%), and test (10%) sets.

4.2 Exploratory Data Analysis

1. **Class Distribution Analysis:** We verified the number of images per emotion class to check for imbalances in class representation. If some of the classes have significantly more images than others, data imbalance might affect model performance, and augmentation may be needed (Fig. 1 and 2)

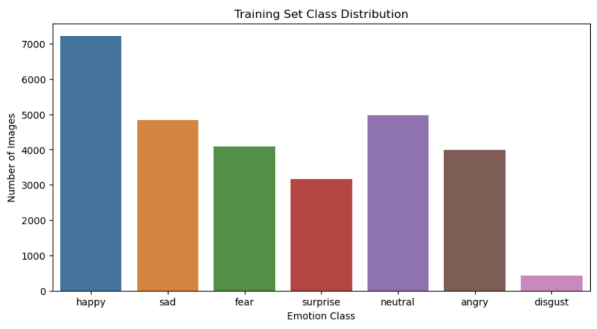

Figure 1: Training Set Class Distribution

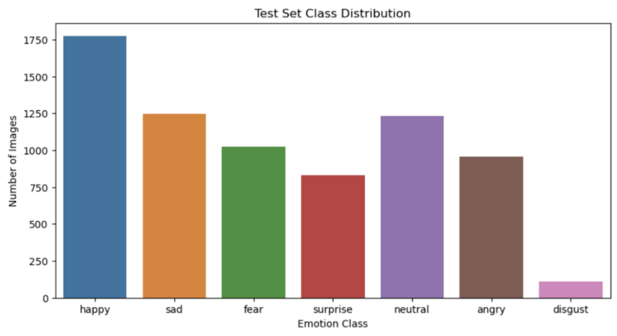

Figure 2: Test Set Class Distribution

2. **Sample Visualization:** Some images may be blurry or low contrast and some emotions might be visually similar. To help assess potential noise, image quality, and variations, we displayed random images from each class to improve our classification (Fig. 3)

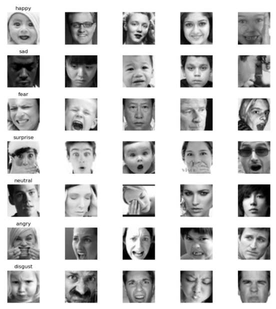

Figure 3: Sample Visualization of the FER-2013 Dataset

3. **Confusion Matrix:** We implemented a confusion matrix to help identify which emotions are commonly misclassified and highlight which emotion pairs are difficult to differentiate. High confusion between emotions suggests that more training data is needed. We analyzed this in the results section (Fig. 3).

5 Model

5.1 Custom CNN Architecture

Our custom CNN is tailored for the FER-2013 dataset and follows this architecture:

- **Input Layer:** 48x48 grayscale image.
- **Convolutional Layers:** Three convolutional layers using 32, 64, and 128 filters respectively, each with a 3×3 kernel and ReLU activation.
- **Pooling Layers:** Max pooling (2x2) window is applied after each convolutional block.
- **Dropout:** 0.5 for regularization–reducing overfitting.
- **Fully Connected Layers:** Dense layer with 256 units and ReLU activation before final classification.
- **Output Layer:** Softmax layer outputs probability distributions over the seven emotions categories.

5.2 Training Procedure

- **Loss Function:** Categorical cross-entropy.

- **Optimizer:** Adam with a learning rate of 0.001.
- **Batch Size:** 64.
- **Epochs:** 30 (with early stopping based on validation performance to prevent overfitting).
- **Evaluation Metrics:** Accuracy, precision, recall, and F1-score are monitored during training, and training curves are generated to assess convergence.

6 Results

To evaluate the model, we observed the confusion matrices to visualize misclassification among emotion categories, accuracy, and loss graphs to track model training progression convergence, and quantitative metrics (precision, recall, F1-score) on the test set.

Within the training directory, the data was further split into 2,968 images for training and 5,741 images for validation. The test directory contained approximately 7,178 images across the seven emotion classes.

The classification report provided the weighted average metrics:

$$\text{Precision: } 0.58, \quad \text{Recall: } 0.59, \quad \text{F1-score: } 0.57.$$

Below is the figure illustrating the confusion matrix

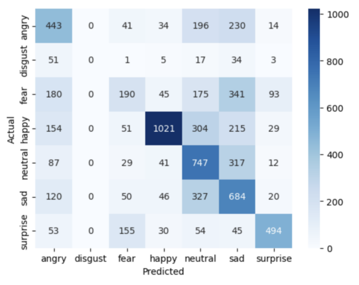

Figure 4: Confusion Matrix on the FER-2013 Dataset

6.1 Observations from the Confusion Matrix

The custom CNN evaluated on the FER-2013 dataset achieved an overall accuracy of 0.58. **Happy** is the best-recognized class with 1021 correct classifications despite some being confused with **Neutral** (304) and **Sad** (215). In contrast, **Disgust** is severely underrepresented with only 3 correct classifications and suffers from very low recall—likely due to a limited number of training samples for that category. It is occasionally misclassified as **Surprise**. **Angry**, correctly identified 443 times, is often mistaken for **Neutral** (196), **Sad** (230), or **Happy** (39), likely due to shared features like furrowed brows. Similarly, **Fear** (190 correct) is commonly confused with **Sad** (341) or **Neutral** (175), while **Neutral** (747 correct) is misclassified as **Sad** (317). This suggests subtle facial cues may be interpreted as slight happiness or sadness. **Sad** correctly classified 684 samples and **Surprise** correctly classified 494 samples often mistaking **Fear** (155) and sometimes **Angry** (53), which may reflect similarities in wide-eyed expressions. One way this model could be enhanced in the future is with a deeper network containing class weighting to address biases or more sophisticated feature extraction methods to better separate these closely related emotional states.

7 Conclusions and Discussion

This project demonstrates the effectiveness of a custom CNN in classifying facial emotions from the FER-2013 dataset. Our model, designed from scratch, achieves promising results through careful pre-processing and a well-structured architecture. Future work will focus on hyperparameter tuning, deeper error analysis, and exploring modifications to further improve classification performance.

8 Unaccomplished or Incomplete Components

- **Hyperparameter Optimization:** Extensive tuning was constrained by available computational resources, which may have further improved performance.
- **In-depth Error Analysis:** More detailed class-specific error analysis could offer additional insights into misclassification trends.
- **Comparison with a Pretrained VGG16 Model:** Due to scope limitations, we did not perform a comparative analysis with a VGG16-based transfer learning approach, which might have provided valuable benchmarks and further insights into model performance.
- **Higher Accuracy Potential:** The overall accuracy may have been improved further if we had access to faster computational resources, as the duration required to run extensive tests on our current hardware limited our ability to fully explore performance-enhancing configurations.

9 Author Contributions

- **Adam Goldstein:** Responsible for dataset preparation, including resizing, normalization, and data augmentation.
- **Taran Elgin:** Designed, built, and trained the custom CNN model, experimenting with different architectures.
- **Lucie Martin:** Evaluated the model, generated performance metrics, and conducted error analysis.
- **All Authors:** Collaboratively wrote and revised the final report.
- **GitHub Link:** `https://github.com/AdamGold19/Math156`

10 Acknowledgments

We would like to thank our course instructor for teaching us about convolutional neural networks.

References

[1] IBM. Convolutional Neural Networks. Retrieved from `https://www.ibm.com/think/topics/convolutional-neural-networks`

[2] GeeksforGeeks. Math Behind Convolutional Neural Networks. Retrieved from `https://www.geeksforgeeks.org/math-behind-convolutional-neural-networks/`

[3] O'Shea, K., & Nash, R. *An Introduction to Convolutional Neural Networks.*

[4] GeeksforGeeks. *OpenCV Python Tutorial.* GeeksforGeeks, 2025. Retrieved from `https://www.geeksforgeeks.org/opencv-python-tutorial/`

[5] TensorFlow. *Image Classification Tutorial.* TensorFlow, 2025. Retrieved from `https://www.tensorflow.org/tutorials/images/classification`

[6] Sambare, M. *FER-2013 Dataset.* Kaggle, 2025. Retrieved from `https://www.kaggle.com/datasets/msambare/fer2013`