Facial Expression Recognition Using CNN on FER2013 Dataset

Nguyen Thien Hao, Thai Trung Hieu and Pham Quang Hieu

June 23, 2025

Abstract :

This paper presents a deep learning-based approach for facial expression recognition using the FER-2013 dataset. A Convolutional Neural Network (CNN) model is developed and trained with a focus on handling key challenges in the dataset such as class imbalance, low image resolution, and inter-class ambiguity. To enhance robustness and prevent overfitting, preprocessing techniques such as normalization, one-hot encoding, and a targeted data augmentation pipeline are applied. The model's performance is evaluated using accuracy, loss, F1-score, and a confusion matrix. The experimental results demonstrate competitive classification accuracy, and the analysis highlights both the model's strengths and limitations, suggesting potential directions for future work.

Index Terms: Facial expression recognition, FER-2013, CNN, data augmentation, deep learning, computer vision.

1 Introduction

Emotion recognition is a branch field of Artificial Intelligence (AI), in particular Computer Vision and Natural Langue Processing, with the goal detecting and classifying human emotions through signals such as: facial expression, speed tone, text segment, gesture, heart rate,... The objective of this research is to detect human emotions through input datasets. Specifically, the system aims to classify human emotions into seven categories: Happy, Sad, Angry, Surprised, Fear, Disgust, and Neutral. Emotion recognition has a wide range of applications in various fields. It can be utilized in customer care to better understand and respond to users' emotional states. In smart education, emotion detection helps personalize learning experiences. It also plays an important role in medical diagnosis by interpreting patients' emotional responses. Additionally, it enhances user experience in virtual reality games by adapting the environment based on emotional feedback.

2 Background

2.1 Computer Vision

Computer Vision is a branch field of AI, researching how computer can "see", "understand" and "analyze" like a human. Dataset of Computer Vision is usually images, videos, frames. The purpose is face recognition, object detect, classify image,... In contrast, general purpose AI algorithms span many other domains beyond vision, including natural language processing (NLP), speech recognition, recommendation systems, and game playing. Many of these traditional AI models are designed to handle structured data (such as text or numerical features) and are not wellsuited for processing complex visual inputs with spatial dependencies. Using conventional models like Multi-Layer Perceptrons (MLPs) to analyze image data is often ineffective, as they do not leverage the spatial structure of images. To address this limitation, more advanced techniques in Computer Vision have been developed—most notably, Convolutional Neural Networks (CNNs), a deep learning architecture specifically designed for image-based tasks. CNNs play a central role in most modern Computer Vision applications due to their ability to automatically and efficiently extract meaningful features from visual data. However, the internal structure and working principles of CNNs will be discussed in more detail in a later section.

2.2 Convolution Neural Network (CNN)

A Convolutional Neural Network (CNN) is a specialized deep learning model designed to process spatially structured data, most notably images. Unlike traditional neural networks such as Multi-Layer Perceptrons (MLPs), CNNs do not fully connect every neuron between layers. Instead, they use convolution operations to scan small local regions of the input image. Each convolution is performed using a kernel (or filter) — a small matrix (e.g., 3×3 or 5×5) — that extracts meaningful features such as edges, corners, or textures.

One of CNN's major strengths is its ability to exploit the spatial locality in data. Rather than looking at the entire image at once like an MLP, CNN processes local regions, mimicking how humans perceive visual information from local details to global understanding. In addition, CNNs share weights: the same kernel is applied across the entire image, allowing the network to detect the same feature in multiple locations. This not only improves generalization but also drastically reduces the number of learnable parameters.

To illustrate the efficiency in terms of parameter count, consider an input image of size 32×32×3 (width, height, and RGB channels). If fed into a fully connected MLP layer with 100 hidden neurons, the number of weights would be:

- 32×32×3×100 = 307,200 weights, since each neuron is connected to every pixel.

By contrast, a CNN layer with 10 convolutional kernels of size 3×3×3 would only require:

- 10 × (3×3×3) = 270 weights.

The difference is dramatic. CNNs require thousands of times fewer parameters, making them faster to train, less prone to overfitting, and more data-efficient. Thanks to these features, CNNs are the dominant architecture in computer vision applications such as image classification, object detection, image segmentation, and tasks like facial recognition, license plate reading, and handwriting recognition.

In summary, CNNs significantly outperform MLPs for image-related tasks by leveraging convolution and weight sharing. They offer greater scalability, flexibility, and representational power, making them central to many modern artificial intelligence applications.

3 Methods

3.1 Dataset and Preprocessing

3.1.1 Dataset Description and Challenges

The FER-2013 (Facial Expression Recognition 2013) dataset, introduced in the ICML 2013 Challenge, serves as the benchmark for this study. It comprises a total of 35,887 grayscale facial images, each with a spatial resolution of 48×48 pixels, categorized into seven discrete emotion classes: *Angry*, *Disgust*, *Fear*, *Happy*, *Sad*, *Surprise*, and *Neutral*.

The FER-2013 includes a training set with 28,709 samples and a validation set with 7178 samples.[1] Despite its popularity, FER-2013 presents several intrinsic limitations that pose significant challenges for facial expression recognition models:

- **Class imbalance:** The dataset exhibits a highly skewed class distribution, with classes such as *Disgust* being significantly underrepresented. This may lead to biased learning and degraded performance on minority classes.
- **Low image resolution:** The limited spatial resolution (48×48 pixels) constrains the availability of fine-grained facial features, making it more difficult to distinguish subtle emotions.
- **High intra-class variability:** Within-class variations arise from differences in lighting, head pose, background, occlusions, and subject demographics, which introduce noise and increase the learning complexity.
- **High inter-class similarity:** Certain expression pairs, such as *Fear* vs. *Surprise*, or *Neutral* vs. *Sad*, share overlapping visual patterns, which increases classification ambiguity.
- **Label noise:** The dataset was annotated via crowdsourcing, and some labels are subjectively ambiguous or inconsistent, introducing additional uncertainty into the learning process.

These characteristics make FER-2013 a realistic yet highly challenging benchmark, suitable for evaluating the robustness and generalization capabilities of CNN-based emotion recognition systems.

3.1.2 Preprocessing and Data Augmentation

To mitigate the aforementioned limitations and enhance model generalization, a set of preprocessing and data augmentation strategies was employed:

Normalization All image pixel intensities were rescaled from the integer range $[0, 255]$ to the floating-point range $[0, 1]$ via linear normalization. This standardization facilitates faster and more stable convergence during gradient-based optimization by ensuring consistent input magnitudes across the dataset.

[1] https://www.kaggle.com/datasets/msambare/fer2013

Label Encoding The emotion class indices were converted into one-hot encoded vectors, enabling compatibility with categorical cross-entropy loss and softmax-based multi-class classification frameworks.

Shuffling and Prefetching The dataset was shuffled before batching to improve sample diversity per iteration. TensorFlow's `AUTOTUNE` prefetching was enabled to asynchronously prepare data during GPU computation, thereby optimizing training throughput.

Data Augmentation To increase data diversity and combat overfitting, a data augmentation pipeline was introduced during training. Given the low resolution and variability of facial expressions in FER-2013, augmentation acts as a critical form of regularization, simulating real-world appearance variations. The following transformations were applied:

- **Random horizontal flipping:** Captures facial symmetry and accounts for mirrored expressions.
- **Random rotation (±10°):** Introduces pose variation, reflecting natural head tilts and off-angle captures.
- **Random zooming (±10%):** Emulates varying camera distances and facial scales.

Figure 1 illustrates the visual effect of data augmentation applied to a sample from the FER-2013 dataset. The original image (right) shows a frontal facial expression in its raw grayscale format. In contrast, the augmented image (left) includes transformations such as zooming, horizontal flipping, and rotation.

(a) Augmented Image **(b)** Original Image

Figure 1. Visual comparison between an original image and its augmented variant. The augmentations introduce spatial transformations that enhance generalization by simulating real-world variability.

Unlike datasets with clean, high-resolution images, FER-2013 suffers from numerous real-world artifacts—ranging from inconsistent illumination to ambiguous facial poses and class imbalance. In this context, data augmentation is not merely a best practice but a necessity. By exposing the model to stochastic variations of the same image during training, augmentation enables the network to develop robustness against overfitting and over-reliance on specific pixel patterns. This is especially beneficial for distinguishing visually similar expressions (e.g., *Fear* vs. *Surprise*) or improving performance on underrepresented classes.

These augmentations are especially motivated by the inherent variability in the FER-2013 dataset, where facial expressions are captured under diverse and uncontrolled conditions — including non-frontal poses, tilted head angles, and inconsistent

alignment. By introducing geometric transformations such as rotation, zooming, and horizontal flipping, the model is exposed to a wider distribution of possible facial configurations that better reflect real-world scenarios. This encourages the network to learn pose-invariant and robust features, rather than overfitting to canonical, front-facing faces. Such regularization is particularly beneficial when distinguishing between visually similar emotions (e.g., *Surprise* vs. *Fear*), where subtle geometric cues play a significant role.

3.2 Model Architecture

In Figure 2, the model is based on the VGG model architecture, which has an increased number of kernels (3x3) and reduced sizes of feature maps over blocks for effective feature extraction. In addition, some other layers are added to address dataset-specific challenges.

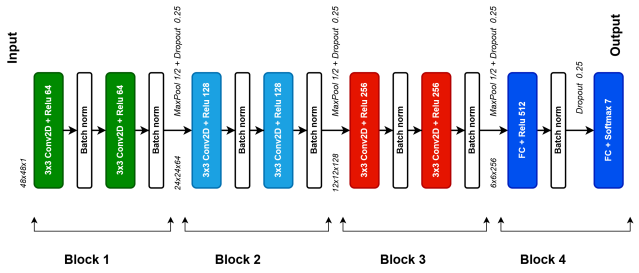

Figure 2. Demonstration of Facial Expression Recognition model.

Batch normalization layers are applied after each layer to reduce internal covariate shift. Therefore, it helps the model partially avoid overfitting and improves training stability. Furthermore, while overfitting has been partly handled through data augmentation, lightweight dropout layers are still added after each convolutional block to further mitigate this issue. Besides, the dense layers in the final, which play a crucial role in classifying features passed from the previous block, are also followed by dropout to enhance generalization.

3.3 Training Setup

3.3.1 Loss Function

In the provided code for training an emotion recognition model, the loss function used is *categorical_crossentropy*. The reason for using *categorical_crossentropy* is that it is well-suited for multi-class classification problems where the target labels are one-hot encoded (each label is represented as a vector with a single element set to 1 and the rest set to 0). Additionally, the model's output layer uses the softmax activation function, which outputs a probability distribution over the 7 emotion classes. Therefore, *categorical_crossentropy* is the appropriate choice to measure the difference between the predicted probability distribution and the actual distribution of the true labels.

3.3.2 Optimizer

In training the emotion recognition model, the optimizer used is the **Adam (Adaptive Moment Estimation)** optimizer to update the model's weights. **Adam** is one of the most popular and effective optimization algorithms in deep learning. Specifically, Adam automatically adjusts the learning rate for each parameter by using estimates of the first and second moments of the gradients, which helps the model converge faster and more stably. Compared to other optimizers such as **SGD (Stochastic Gradient Descent)** or **Adagrad**, **Adam** demonstrates better adaptability to large datasets and sparse gradients, and it typically requires less hyperparameter tuning. Therefore, using **Adam** simplifies the training process while still achieving high accuracy.

3.3.3 LRreduce

During the training process, I used the **ReduceLROnPlateau** callback to automatically adjust the learning rate in order to optimize convergence. Specifically, this callback monitors the val_loss value and reduces the learning rate by a factor (in this case, 0.5) if the model does not show improvement on the validation loss after a certain number of epochs (patience = 3). This mechanism helps prevent the model from getting stuck in suboptimal states due to a high learning rate and allows for slower, more stable convergence as the model approaches a minimum. As a result, **ReduceLROnPlateau** contributes to more efficient training without requiring manual tuning of the learning rate during the process.

3.3.4 Early Stopping

During the training process, I used the **EarlyStopping** callback to automatically stop training when the model stopped improving. Specifically, the training would halt if the validation loss (val_loss) did not decrease for 50 consecutive epochs. Additionally, by setting **restore_best_weights=True**, the model automatically reverts to the weights from the epoch where it achieved the best performance on the validation set. This approach helps prevent overfitting and saves training time by avoiding unnecessary epochs once the model has converged.

4 Result

To thoroughly assess the model's performance, we employ a multi-faceted evaluation strategy, including visual analysis of accuracy and loss progression, class-wise performance via the confusion matrix, and supporting metrics such as the F1-score. These collectively help us understand not only how well the model fits the data, but also how well it generalizes to unseen samples.

4.1 Accuracy and Loss Curves

Figures 3 and 4 depict the training and validation accuracy and loss throughout the training process. The model achieves consistent performance, with training accuracy steadily increasing and eventually saturating around 72%. Validation accuracy stabilizes slightly below that, around 66–67%, which is a strong result for the challenging FER-2013 dataset.

The loss curve corroborates this trend. Training loss continually decreases and eventually converges, while validation loss exhibits a clear plateau without significant divergence—suggesting no major overfitting. Notably, the gap between training and

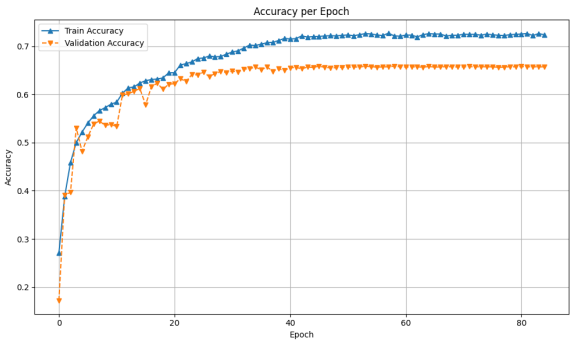

Figure 3. Training and validation accuracy across epochs.

validation performance, although present, is acceptable given the dataset's noise and complexity.

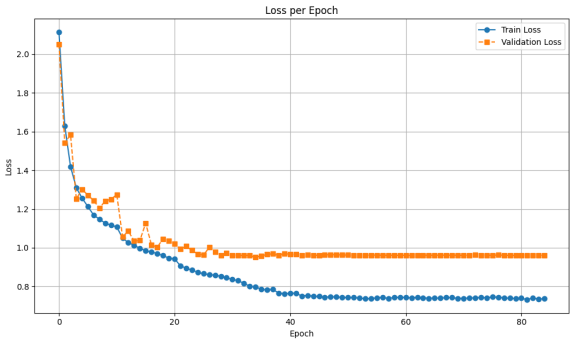

Figure 4. Training and validation loss across epochs.

These trends indicate the model has learned to extract generalizable facial expression features rather than memorizing specific training samples. Techniques such as dropout, batch normalization, and data augmentation proved crucial in achieving this stability.

4.2 Confusion Matrix and Class-wise Analysis

Figure 5 illustrates the confusion matrix on the validation set, offering detailed insights into the class-wise performance of the model. The matrix demonstrates a strong diagonal dominance, particularly for well-represented classes such as *Happy*, *Sad*, and *Neutral*, indicating reliable classification behavior in these categories.

However, notable misclassifications are observed between several expression pairs, most prominently between *Fear* and *Surprise*, as well as between *Angry* and visually adjacent expressions. These confusions are consistent with both the visual similarity of facial expressions and the class imbalance inherent in the FER-2013 dataset.

Key Observations:

- **Happy** achieves the highest classification accuracy with 1,524 correctly predicted samples. This can be attributed to both its strong visual features (e.g., smiling mouth, elevated cheeks) and its prevalence in the dataset.

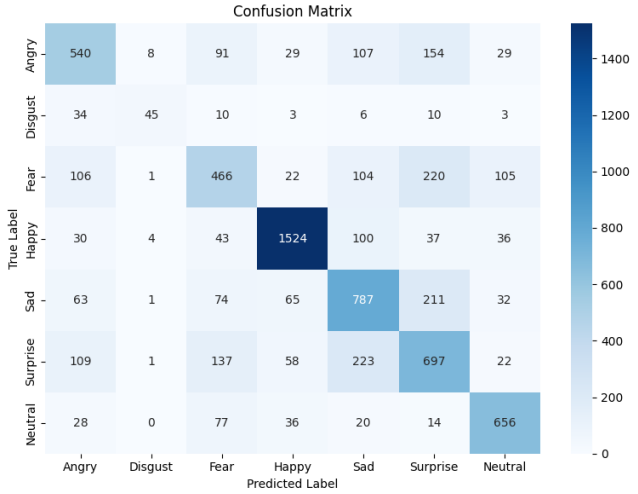

Figure 5. Confusion matrix on validation data. Rows represent true labels; columns represent predicted labels.

- **Fear** is frequently misclassified as *Surprise* (220 instances) and *Neutral* (105 instances), indicating the model's limited ability to distinguish nuanced emotional expressions under constrained image quality.
- **Surprise** is misinterpreted as *Fear* (137) and *Sad* (223), highlighting the model's challenge in separating high-arousal emotional states that often exhibit overlapping facial cues (e.g., wide eyes, raised eyebrows).
- **Disgust** yields the lowest recognition performance, with only 45 correct predictions. The model tends to misclassify it as *Angry*, *Fear*, or *Sad*, which is unsurprising given its scarce representation and subtle facial characteristics.
- **Angry** often overlaps with *Fear* (91) and *Surprise* (154), potentially due to shared upper-face features such as intense gaze or eye widening, especially under low-resolution grayscale conditions.

Overall Performance The confusion matrix confirms that the model has learned meaningful patterns, particularly for dominant and visually distinct classes. Nevertheless, ambiguity remains in subtle or underrepresented emotions, primarily due to dataset constraints such as grayscale input, low resolution, pose variation, and class imbalance. Despite these challenges, the model exhibits competitive performance for a single-stream CNN. The strong diagonal structure of the confusion matrix and relatively controlled off-diagonal noise demonstrate a well-generalized classifier. Future improvements may include class reweighting strategies, integration of facial alignment or landmark features, and ensemble learning to enhance robustness across hard-to-distinguish emotion classes.

4.3 Classification Report

Table 1 presents the detailed classification metrics, including precision, recall, and F1-score, for each emotion class. Overall, the model achieves a macro-averaged F1-score of **0.692**, reflecting balanced performance across multiple categories despite the inherent challenges of the FER-2013 dataset.

Table 1
Classification report by class.

Class	Precision	Recall	F1-score	Support
Angry	0.6758	0.6652	0.6705	409
Disgust	0.7500	0.5306	0.6225	49
Fear	0.6224	0.5314	0.5737	412
Happy	0.8251	0.8955	0.8588	720
Sad	0.6486	0.6503	0.6495	498
Surprise	0.7879	0.8033	0.7955	445
Neutral	0.6676	0.6841	0.6757	556

The classification results indicate that the model exhibits strong discriminative power for certain emotion classes while struggling with others:

- **Happy** and **Surprise** achieved the highest F1-scores (0.8588 and 0.7955, respectively), owing to their high sample counts and salient facial features (e.g., smiling, wide eyes).
- **Disgust** and **Fear**, on the other hand, recorded the lowest F1-scores. These classes suffer from both data scarcity (e.g., only 49 instances of *Disgust*) and high inter-class similarity with expressions such as *Angry* or *Surprise*, making them inherently harder to classify.
- The relatively balanced F1-scores for *Angry*, *Sad*, and *Neutral* suggest that the model has learned generalized representations even for mid-frequency classes.

These outcomes reaffirm known challenges in facial expression recognition, particularly on in-the-wild datasets like FER-2013 where low resolution, grayscale input, pose variation, and annotation noise hinder optimal performance.

Overall, while the proposed CNN architecture performs competitively within the constraints of FER-2013, these results underline the importance of architectural innovations and better data strategies for advancing facial expression recognition.

5 Discussion

Throughout the previous sections, a facial expression recognition system has been successfully developed. However, several limitations remain at various stages of the pipeline, which should be addressed to further improve and refine the model. These include:

Class imbalance mitigation As previously mentioned, the dataset is imbalanced, which leads to biased learning and inferior results on less common labels. The pipeline has not yet addressed this problem. Therefore, incorporating data-level strategies, such as oversampling, and undersampling, is recommended to better balance the class distribution. However, it may still limit generalization due to insufficient sample diversity, which prevents the model from learning representative features effectively. Eventually, data variety improvement is a significant aspect that should be considered.

Loss of partial information In the architecture, drop-out layers follow each block, even convolutional blocks. However, this regularization method was originally designed for fully connected layers where data is flattened. Therefore, the use of drop-out for convolutional blocks may disrupt spatial structures. Further research should try another variant of drop-out like spatial drop-out, which deactivates entire feature maps during training, rather than individual units.

Optimizer selection Adam was selected for the implementation due to its fast convergence speed. However, its generalization performance remains suboptimal. [2] Further work should consider SGD with momentum and carefully tuned hyperparameters to enhance inference. [3] A hybrid method is also one of the notable strategies: Adam optimizer could be applied during initial training epochs to accelerate convergence and followed by SGD in later stages to improve generalization.

In summary, while the proposed system shows promising results, several areas such as data imbalance, generalization capability, and optimizer choice require further attention. Addressing these limitations can significantly enhance the model's performance and robustness in real-world scenarios.

6 Conclusion

This study presented a CNN-based facial expression recognition system trained on the FER-2013 dataset, addressing key challenges such as class imbalance, low resolution, and expression ambiguity. Through careful preprocessing, data augmentation, and architectural regularization, the model achieved competitive performance with strong generalization capacity. While certain limitations remain, particularly in handling underrepresented and visually similar emotions, the proposed approach establishes a solid baseline for future improvements. Incorporating advanced training strategies and richer datasets may further enhance real-world applicability in emotion-aware systems.

Appendix

Here is the source code.

[2] Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B. (2017, December). The marginal value of adaptive gradient methods in machine learning. Advances in Neural Information Processing Systems, 30, 4151–4161. arXiv:1705.08292

[3] Keskar, N. S., Socher, R. (2020, October). Do you need an adaptive optimizer for training deep neural networks? arXiv:2010.13440

Reference

1. https://www.kaggle.com/datasets/msambare/fer2013
2. Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B. (2017, December). The marginal value of adaptive gradient methods in machine learning. Advances in Neural Information Processing Systems, 30, 4151–4161. arXiv:1705.08292
3. Keskar, N. S., Socher, R. (2020, October). Do you need an adaptive optimizer for training deep neural networks? arXiv:2010.13440