**A CNN-BASED APPROACH TO EMOTION RECOGNITION USING FER2013**

Mariia Kalianova, Danylo Tovpyshko

La Sapienza University in Rome

# Abstract

This paper presents a convolutional neural network (CNN)-based method for facial emotion recognition using the FER2013 dataset. The goal is to classify facial expressions into one of seven basic emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. The network is trained and tested using PyTorch, and deployed via a graphical interface built with Gradio and OpenCV. This research demonstrates the feasibility of deploying lightweight models for real-time emotion recognition in everyday applications.

# Index Terms

Emotion recognition, CNN, FER2013, PyTorch, facial expressions, Gradio, OpenCV.

## I. Introduction

Facial emotion recognition has gained attention in human-computer interaction and mental health assessment. Recognizing emotions from facial cues in real-time can enable adaptive interfaces, monitoring tools, and educational systems. In this work, we focus on creating a practical emotion recognition system using a self-built CNN trained on FER2013, a widely used public dataset for facial expressions.

## II. Related Work

Prior studies employed deep learning models, especially CNNs, due to their strong performance in image-based tasks. Some works leveraged complex architectures like VGG or ResNet, while others opted for lightweight models suitable for real-time use. Our approach emphasizes simplicity and interpretability, targeting deployability rather than marginal accuracy gains.

## III. Methodology

## A. Dataset

The FER2013 dataset includes 35,887 grayscale 48x48 facial images categorized into seven emotion classes. The dataset is split into training, public test, and private test sets. Preprocessing steps included: Random horizontal flip, Random rotation, Tensor conversion, Normalization. These techniques help the model generalize better and reduce overfitting.

## B. Model Architecture

We implemented a custom CNN named EmotionRecognitionCNN using PyTorch. It contains multiple convolutional layers with ReLU activations, batch normalization, dropout, and max pooling, followed by fully connected layers for classification.

## C. Training

The model was trained using cross-entropy loss and the Adam optimizer. Training was conducted for 10 epochs to ensure convergence while avoiding overfitting.

## IV. Results

The model achieved 58.86% accuracy on the FER2013 test set. This performance is competitive given the dataset limitations. The final model (emotion\_model3\_58.pth) showed robustness, though confusion between 'sad' and 'neutral' was observed.

## V. Deployment and GUI

We developed a real-time graphical interface using Gradio. It supports image and video uploads and performs live webcam inference. The system runs efficiently on standard machines and is suitable for educational or demonstrative use. Emotion detection from webcam input is powered by OpenCV.

## VI. Conclusion and Future Work

This work demonstrates that a simple CNN with proper preprocessing can deliver reliable emotion recognition. Future improvements may include: Multi-modal input (e.g., audio), Better model interpretability, Expanded emotion datasets with more granularity.

## VII. Challenges and Limitations

1. Class Imbalance: FER2013 has unbalanced class distribution, causing misclassification in underrepresented classes like 'disgust' and 'fear'.  
2. Low-Resolution Grayscale Images: Images are 48x48 pixels in grayscale, limiting model learning potential.  
3. Real-Time Constraints: Balancing inference speed with accuracy was challenging, especially without GPU.  
4. Emotion Ambiguity: Emotions like 'sad' and 'neutral' are hard to distinguish even for humans.  
5. Hardware Limitations: Training was conducted on a CPU-only machine.  
6. Face Detection Issues: Haar Cascades in OpenCV sometimes failed to detect faces.  
7. Gradio Webcam Failures: Webcam integration was occasionally unreliable. That was not possible to implement webcam in Gradio due to environmental issues. The solution was to separate main functionality (uploading photos and videos) and webcam functionality in two separate files.