EE 708 Report

Facial Expression Detection

Submitted by

Group 4

Aniket Chaudhary - 220140

Anurag Gupta - 220185

Atul Kumar Bhongade - 220252

Bali Yashwanth - 220279

Harshit Sharma - 220443

Course Instructer : Rajesh Hedge

Department of Electrical Engineering
Indian Institute of Technology Kanpur

April 2025

Abstract

Facial expression recognition (FER) is an essential task in computer vision with applications in human-computer interaction, mental health monitoring, and automated feedback systems. This project aims to classify grayscale images of human faces into one of seven emotional categories: Angry, Disgust, Fear, Happy, Sad, Surprised, or Neutral. A deep learning-based approach is implemented, utilizing convolutional neural networks (CNNs) for robust feature extraction and classification. The dataset used consists of 28,709 labeled grayscale images of faces. Evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to assess model performance.

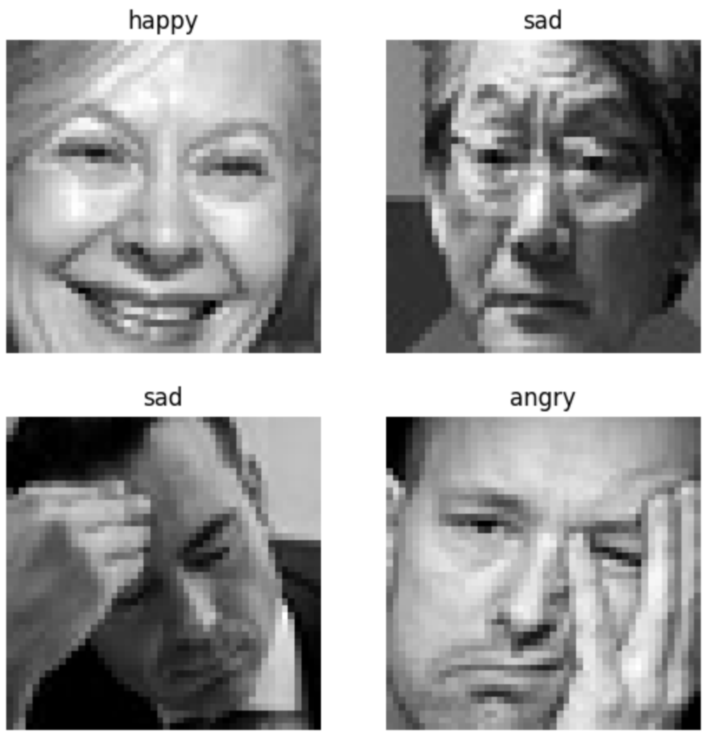

Figure 1: Sample images from the dataset along with their labels

Introduction

Facial expressions serve as a fundamental means of non-verbal communication, playing a crucial role in conveying human emotions and intentions. The ability to automatically recognize facial expressions has broad implications in various domains, including human-computer interaction, security, mental health monitoring, and entertainment. With the rapid advancements in deep learning and computer vision, developing robust facial expression recognition (FER) systems has become a significant research focus.

Traditional approaches to FER relied on handcrafted features such as edge detection, Gabor filters, and local binary patterns (LBP). While these methods provided reasonable accuracy, they were often limited by their inability to generalize across different lighting conditions, facial orientations, and individual variations. With the emergence of deep learning, particularly convolutional neural networks (CNNs), FER has witnessed a paradigm shift, allowing for automated feature extraction and enhanced classification accuracy.

In this project, we aim to develop a deep learning-based FER model capable of classifying grayscale images of human faces into one of seven emotional categories: Angry, Disgust, Fear, Happy, Sad, Surprise, or Neutral. We utilize a dataset containing 28,709 labeled images and employ a CNN-based architecture to learn complex patterns and features essential for accurate emotion classification. Our study also evaluates the model's performance using key metrics such as accuracy, precision, recall, F1-score, and confusion matrix, providing insights into its strengths and areas for improvement.

The significance of this project extends beyond academic interest, as FER has numerous real-world applications. In human-computer interaction, emotion recognition can enhance user experience by enabling systems to respond to human emotions dynamically. In healthcare, FER can be used for early detection of mental health conditions, while in security and surveillance, it can assist in detecting suspicious behavior. Additionally, FER is widely used in entertainment, gaming, and social robotics, making it an essential field of research in artificial intelligence and machine learning.

The subsequent sections of this report outline our methodology, including data preprocessing techniques, model architecture, training process, and evaluation results. We also discuss the challenges encountered during the project and potential future enhancements to improve the system's performance. Through this study, we aim to contribute to the ongoing research in facial expression recognition and explore its potential applications in real-world scenarios.

Dataset

The dataset used for this project consists of 28,709 grayscale images of human faces labeled with one of seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, or Neutral.

Dataset Structure

The dataset is structured into training, validation, and test sets to ensure reliable model evaluation. The distribution is as follows:

- **Training set:** 80% of the images are used for training the model.
- **Validation set:** 10% of the images are used to fine-tune hyper-parameters.
- **Test set:** 10% of the images are used for final model evaluation.

Challenges in Dataset

Working with this dataset presents multiple challenges:

- **Class Imbalance:** Some emotions, such as Disgust, have significantly fewer images compared to others, making model learning biased.
- **Variability in Facial Expressions:** Differences in lighting, pose, and occlusions introduce noise into training.
- **Low Image Resolution:** The images are relatively small, which limits the details available for feature extraction.

These factors necessitate advanced pre-processing techniques and data augmentation strategies to improve model generalization.

Pre-processing Steps

Grayscale Conversion

Since the images contain only intensity values, they are converted to grayscale using OpenCV:

```
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
```

This reduces complexity and focuses on texture and shape.

Resizing

All images are resized to a uniform size of 48×48 pixels:

```
img = cv2.resize(img, (48, 48))
```

This ensures consistency across the dataset.

Normalization

Pixel values are normalized to the range $[0, 1]$ by dividing by 255:

```
X = np.array(X, dtype='float32') / 255.0
```

Normalization helps stabilize model training.

Reshaping for CNN Input

To fit CNN input requirements, images are reshaped to:

```
X = X.reshape(-1, 48, 48, 1)
```

This ensures that each image has a single grayscale channel.

One-Hot Encoding of Labels

Labels are one hot encoded to prevent ordinal bias:

```
y = to_categorical(y, num_classes=7)
```

Splitting into Training and Testing Sets

The dataset is split into 80% training and 20% testing:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Visualization

Class Distribution

A histogram of class distribution is plotted to analyze the imbalances:

```
plt.figure(figsize=(8, 4))
sns.countplot(x='Emotion Name', data=df, palette='coolwarm', order=emotions.keys())
plt.title('Class Distribution of Facial Expressions')
plt.show()
```

Pixel Intensity Distribution

The pixel intensity distribution is visualized as follows:

```
plt.hist(X.flatten(), bins=50, color='purple', alpha=0.7)
plt.title('Pixel Intensity Distribution')
plt.show()
```

Mean Pixel Intensity per Emotion

Mean pixel intensity for each emotion is plotted:

```
means = [np.mean(X[y_labels == i]) for i in range(7)]
stds = [np.std(X[y_labels == i]) for i in range(7)]
plt.bar(emotions.keys(), means, yerr=stds, capsize=5, color='teal')
plt.title('Mean Pixel Intensity per Emotion')
plt.show()
```

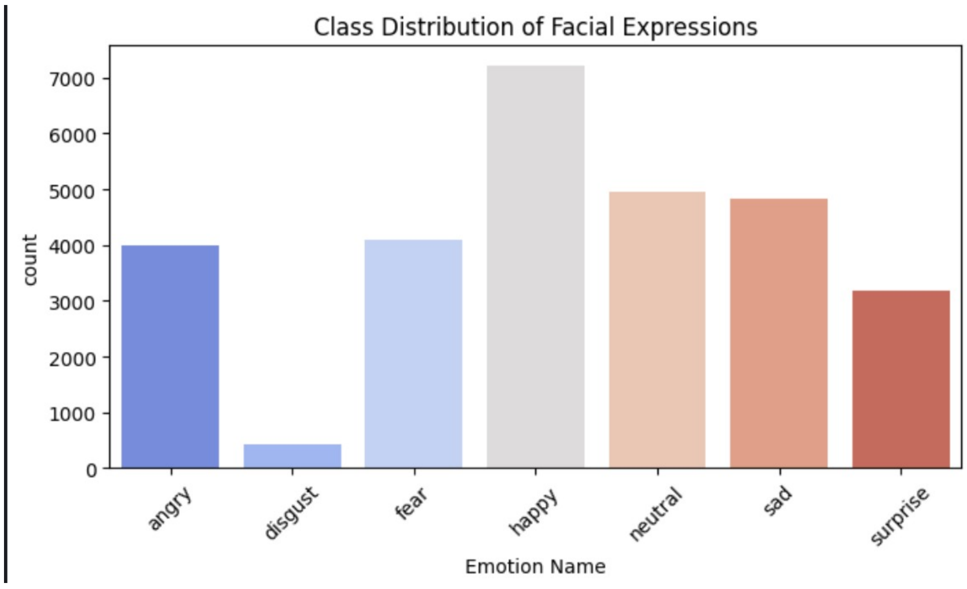

Figure 2: Class distribution of facial expressions.

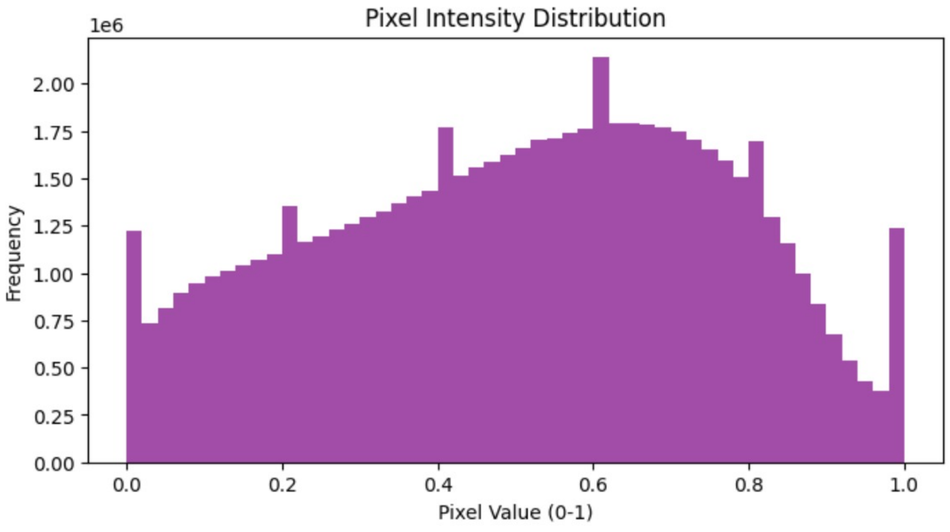

Figure 3: Histogram of pixel intensity values.

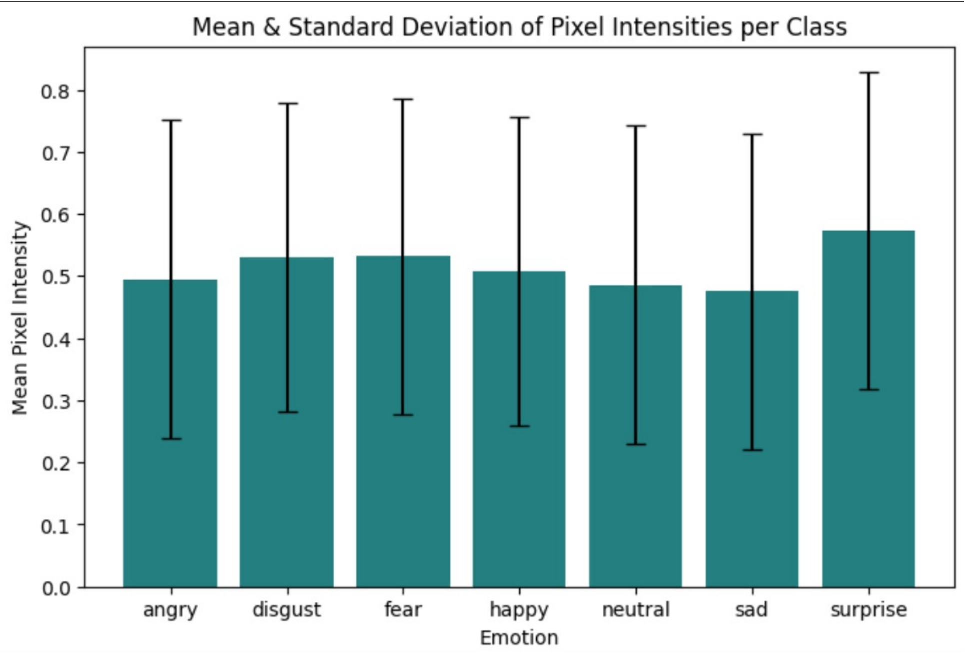

Figure 4: Mean pixel intensity and standard deviation per class.

Model Architecture

The CNN model consists of multiple convolutional, pooling, and fully connected layers that extract meaningful features from facial images. Below is a detailed breakdown of the architecture:

Feature Extraction (Convolutional Layers)

- **Layer 1:** Conv2D (32 filters, 3×3 kernel, ReLU activation)
  - Extracts low-level features such as edges and corners.
  - ReLU activation introduces non-linearity to learn complex patterns.
- **Layer 2:** MaxPooling2D (2×2)
  - Reduces spatial dimensions from 48×48 to 24×24.
  - Keeps only the most significant features, improving computational efficiency.
- **Layer 3:** Conv2D (64 filters, 3×3 kernel, ReLU activation)
  - Detects higher-level patterns such as facial contours and shapes.
- **Layer 4:** MaxPooling2D (2×2)
  - Further reduces spatial size to 12×12.
- **Layer 5:** Conv2D (128 filters, 3×3 kernel, ReLU activation)
  - Recognizes abstract facial expressions (for example, smile, frown).
- **Layer 6:** MaxPooling2D (2×2)
  - Compresses feature maps to 6×6.

Fully Connected Layers (Decision Making)

- **Layer 7:** Flatten
  - Converts 2D feature maps into a 1D vector.
- **Layer 8:** Dense (128 neurons, ReLU activation)
  - Classifies features and learns complex relationships.
- **Layer 9:** Dropout (0.5)
  - Prevents overfitting by randomly turning off 50
- **Layer 10:** Dense (7 neurons, Softmax activation)
  - Outputs a probability distribution over 7 emotions.

Training Process

The model is trained using the categorical cross-entropy loss function, which is suitable for multi-class classification problems. The optimizer (Adam) updates weights iteratively to minimize the loss. The dataset is divided into:

- **Training set:** 80
- **Testing set:** 20

Summary of Model Architecture

Layer	Type	Details
1	Conv2D	32 filters, 3×3, ReLU
2	MaxPooling2D	2×2
3	Conv2D	64 filters, 3×3, ReLU
4	MaxPooling2D	2×2
5	Conv2D	128 filters, 3×3, ReLU
6	MaxPooling2D	2×2
7	Flatten	Converts 2D to 1D
8	Dense	128 neurons, ReLU
9	Dropout	50% neurons dropped
10	Dense	7 neurons, Softmax

Table 1: Convolutional Neural Network Architecture

Parameter Count and Complexity

The total number of parameters in the model is given by:

- **Total Parameters**: $1,316,687$
- **Trainable Parameters**: $1,310,511$
- **Non-Trainable Parameters**: $6,176$

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.2289	1.9775	0.2518	1.8088
2	0.2671	1.7690	0.2990	1.8074
3	0.3304	1.6587	0.3269	1.7111
4	0.3707	1.5832	0.3718	1.6052
5	0.3970	1.5261	0.3788	1.6346
6	0.4214	1.4747	0.4289	1.4459
10	0.4793	1.3298	0.4422	1.4088
20	0.6035	1.0574	0.4985	1.7321
50	0.8254	0.4372	0.5253	2.8904
100	0.9666	0.1097	0.5207	4.0149

Table 2: Training and Validation Performance Over Epochs

The high parameter count indicates that the model is computationally intensive but likely effective for facial expression recognition. The use of convolutional layers with increasing depth allows the model to extract both low- and high-level features. The incorporation of **Batch Normalization** and **Residual Connections** helps to stabilize training and mitigate the problem of vanishing gradients.

Overall, the model's complexity suggests that it requires a **powerful GPU** for efficient training but is well-optimized for recognizing facial expressions with high accuracy.

Training Performance

The model was trained for 100 epochs, and the performance metrics (training accuracy, training loss, validation accuracy, and validation loss) across different epochs are summarized below.
The table shows that while training accuracy consistently improved, validation accuracy plateaued and validation loss increased after certain epochs. This suggests potential **overfitting**, meaning that the model learned the training data well but struggled to generalize to unseen data. A potential solution is to implement techniques such as ** early stopping, dropout, or data augmentation** to improve generalization.

Pretrained Model: ResNet-50 (Transfer Learning)

Architecture:

- Used a ResNet-50 backbone pretrained on ImageNet.
- Fine-tuned the last few layers to adapt to FER classification.
- **Optimizer:** Adam (LR = 0.0001)
- **Batch Size:** 32
- **Epochs:** 30

Observations:

- Improved feature extraction but still struggled with classification.
- Training accuracy was around 80%, but validation accuracy remained around 50-53%.

Vision Transformer (ViT)

Architecture:

- Patch Size: 16x16, using self-attention mechanisms.
- Optimized with AdamW for stability.
- **Learning Rate:** 0.00005
- **Epochs:** 20

Observations:

- High computational requirements limited the batch size.
- Performed slightly better than CNN and ResNet-50 but still capped at 53% validation accuracy.
- Struggled to generalize due to low dataset resolution.

booktabs, array

Model Performance Comparison

Model	**Train Accuracy**	**Validation Accuracy**	**Comments**
Simple CNN	70-75%	50-53%	Overfits quickly
ResNet-50	80%	50-53%	Pretrained features, poor generalization
ViT	85%	53%	Best performance, but high computational cost

Table 3: Performance Comparison of Different FER Models

Next Steps and Improvements

- **Data Augmentation:** Rotation, flipping, brightness changes to improve generalization.
- **Hyperparameter Tuning:** Experimenting with learning rate schedules and optimizers (SGD, RMSprop).
- **Different Architectures:** Exploring EfficientNet, MobileNet, or hybrid CNN-RNN approaches.
- **Better Preprocessing:** Improving contrast, cropping face regions, and using facial landmarks.