EC6672 : Machine Intelligence Laboratory

Report of mini project

Face Emotion Recognition

Submitted to: Prof. Samit Ari
EC department, Nit Rourkela

Submitted by: Shailesh Kumar
Course: M. tech
Specialization: Signal and image processing
Roll No: 224EC6013

Department of Electronics and Communications Engineering

National Institute of Engineering ,Rourkela

Rourkela, Odisha

Index

Face Emotion Recognition

1. Objective

To develop a facial emotion recognition system using convolutional neural networks (CNN) that classifies human emotions from facial expressions into predefined categories such as Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.

2. Introduction

Emotions are a key component of human interaction. In recent years, the advancement of Artificial Intelligence has enabled machines to understand and interpret human emotions, which has a wide range of applications including healthcare, security, education, and human-computer interaction.

Facial Emotion Recognition (FER) is a subfield of computer vision where an AI model is trained to detect human facial expressions and classify them into emotional categories. In this project, we utilize Convolutional Neural Networks (CNNs) to automatically extract features from facial images and accurately predict the emotion class.

3. Tools and Libraries Used

- Programming Language: Python 3.x
- Libraries and Frameworks:
    - TensorFlow & Keras (Model Development)
    - NumPy, Pandas (Data Handling)
    - Matplotlib, Seaborn (Visualization)
    - Scikit-learn (Evaluation)
    - OpenCV (Image Handling)
    - Jupyter Notebook / Google Colab (Execution Environment)
    - ImageDataGenerator (Data Augmentation)

4. Theory

4.1 Facial Emotion Recognition (FER)

Facial Emotion Recognition (FER) is a subdomain of computer vision and affective computing that aims to detect and classify human emotional states from facial expressions. It involves analysing visual input—typically static images or video frames—to determine the underlying emotional state of the subject. Emotions are categorized into a discrete set of classes: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. Facial expressions are among the most universal and non-verbal methods of conveying emotions. Automated FER systems have found applications in areas such as:

• Human-computer interaction

• Surveillance and security

• Virtual assistants

• Emotion-aware robotics

• E-learning and psychological studies

FER tasks are generally mapped to seven discrete emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral.

4.2 Deep Learning in FER

Earlier approaches involved feature engineering using techniques like HOG, Gabor Filters, and LBP. However, these techniques lack robustness in real-world scenarios.

Deep Learning, especially CNNs, can learn hierarchical feature representations directly from image pixels, making them ideal for FER tasks. CNNs outperform traditional approaches by capturing spatial hierarchies and complex features.

4.3 Convolutional Neural Networks (CNN)

CNNs are a type of deep neural network designed for image data. Their layers include:

Layer	Function
Conv2D	Extract local features (edges, textures)
ReLU	Non-linear activation
Batch Norm	Normalize activations for stability
MaxPooling2D	Reduce spatial dimensions
Dropout	Regularization to prevent overfitting
Global Avg Pooling	Replaces fully connected layers to reduce parameters
Dense Layer	Final decision layer
Softmax Output	Converts outputs to emotion class probabilities

5 Dataset Description

The dataset used for this facial emotion recognition project is based on labelled grayscale images categorized into seven distinct emotional expressions.

The dataset is organized into two main directories:

- Training Set (/train)
- Test Set (/test)

Each of these directories contains seven subfolders, one for each emotion class:

{Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral}

Image Details

- Image Format: Grayscale
- Image Size: 48 × 48 pixels
- Color Channels: 1 (grayscale)
- Data Type: 8-bit image files (usually .jpg)
- Normalization: Pixel values are scaled to the range [0, 1]

Class Distribution: While the number of images per class may vary slightly, the dataset contains thousands of labelled examples, ensuring sufficient samples to train a deep learning model. The data is slightly imbalanced; classes like Happy, Sad, and Angry have more samples compared to Disgust.

6 Methodology

6.1 Data Preprocessing and Augmentation

The dataset was divided into training and testing sets, each containing images grouped by emotion. To enhance generalization and performance, preprocessing and augmentation techniques were applied:

- Normalization: All image pixel values were scaled to the range [0,1] using rescale=1./255.
- Augmentation Techniques Applied to Training Data:
  - Rotation: ±25 degrees
  - Width/Height Shift: ±20%
  - Shearing and Zooming
  - Horizontal Flipping

These augmentations help the model generalize better to new facial expressions and pose variations.

6.2 Model Architecture

The facial emotion recognition model was built using a deep Convolutional Neural Network (CNN) tailored for grayscale facial images. It includes multiple convolutional, pooling, and normalization layers, designed to effectively capture subtle features.

CNN Model Overview:

Layer Type	Description
Input Layer	48×48×1 grayscale images
Conv2D (64 filters)	3×3 kernel, padding='same', ReLU
Batch Normalization	
Conv2D (64 filters)	3×3 kernel, ReLU
Batch Normalization	
MaxPooling2D	2×2
Dropout (0.25)	
Conv2D (128 filters)	3×3, ReLU, BatchNorm, MaxPooling, Dropout
Conv2D (256 filters)	3×3, ReLU, BatchNorm, MaxPooling, Dropout
Global Average Pooling	Reduces dimensionality
Dense (256 units)	Fully connected, ReLU, BatchNorm, Dropout (0.4)
Output Layer	7 neurons (Softmax activation)

Key Strengths of the Architecture

- Deep enough to capture high-level features.
- Regularization: Dropout and Batch Normalization were applied.
- Global Average Pooling helps reduce overfitting.
- High performance in recognizing complex and subtle expressions.

6.3 Hyperparameters and Optimization Techniques

To train the CNN efficiently and avoid overfitting, the following hyperparameters and optimization strategies were used:

Hyperparameters Used

- **Optimizer:** Adam (Learning Rate = 0.0005)
- **Loss Function:** Categorical Cross entropy
- **Batch Size:** 64
- **Epochs:** 40

Callbacks Used

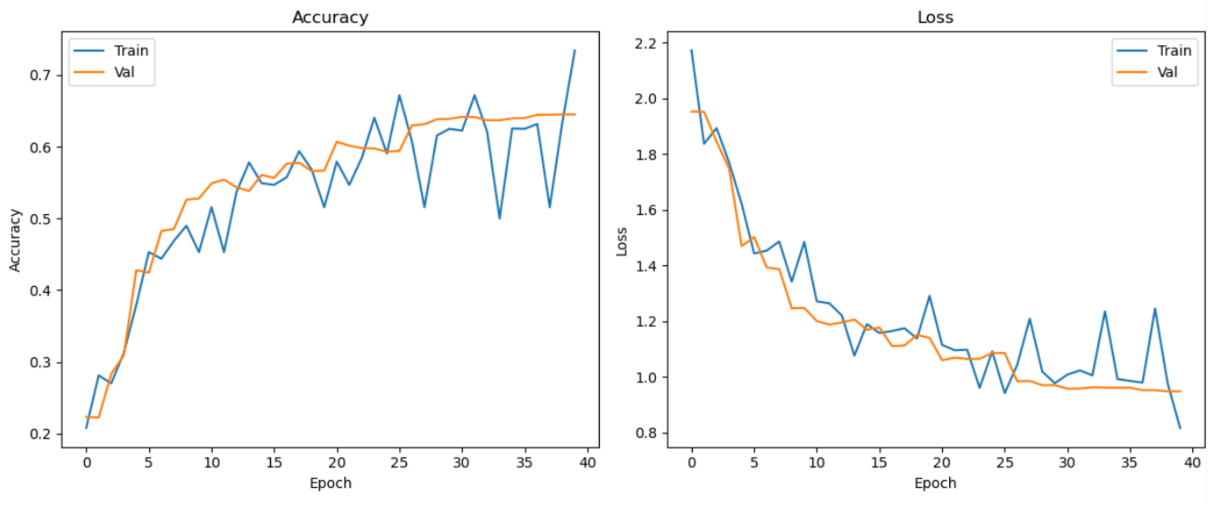
- **EarlyStopping:** Stops training when validation loss does not improve, preventing overfitting.
- **ReduceLROnPlateau:** Reduces the learning rate automatically when a performance plateau is detected.

6.4 **Model Evaluation**

After training, the model's performance was assessed using various evaluation metrics:

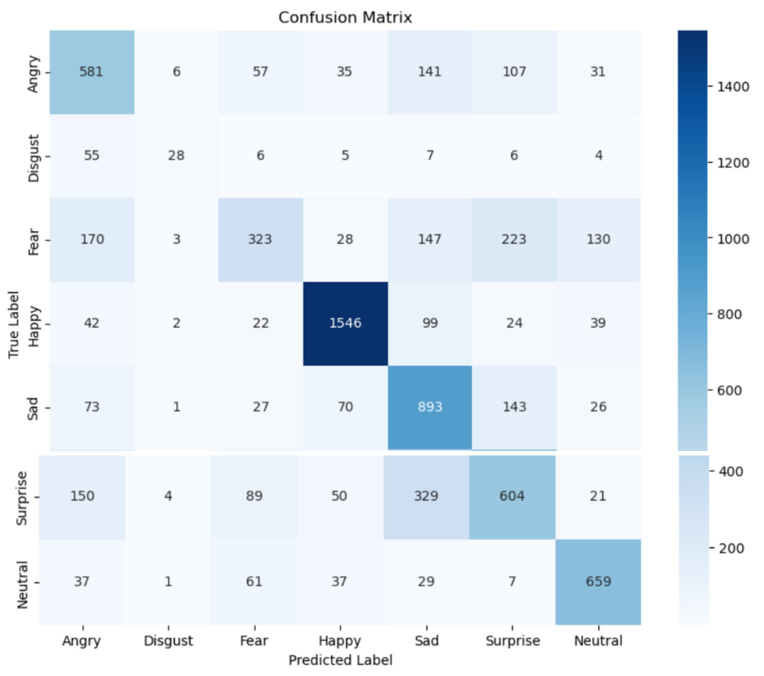
- **Accuracy:** Measures the overall percentage of correctly classified samples.
- **Loss (Categorical Cross entropy):** Quantifies how different predicted probabilities are from the true labels.
- **Classification Report:** Provides per-class **Precision**, **Recall**, and **F1-score**, useful in class imbalance scenarios.
- **Confusion Matrix:** Visualizes how well each emotion is classified, highlighting common misclassifications between similar expressions (e.g., Sad vs. Neutral).

7. Result:



Classification Report:

Emotion	Precision	Recall	F1-Score	Support
Angry	0.52	0.61	0.56	958
Disgust	0.62	0.25	0.36	111
Fear	0.55	0.32	0.40	1024
Happy	0.87	0.87	0.87	1774
Sad	0.54	0.72	0.62	1233
Surprise	0.54	0.48	0.51	1247
Neutral	0.72	0.79	0.76	831
Overall Accuracy			**0.65**	**7178**
Macro Avg	0.63	0.58	0.58	7178
Weighted Avg	0.65	0.65	0.64	7178



8. Discussion and Observations

8.1 Training and Validation Performance

The training and validation accuracy and loss curves reveal several important insights:

- **Accuracy Trend**:
  - Both training and validation accuracy steadily improve over the epochs, reaching close to **70%**.
  - The validation accuracy follows the training curve closely, indicating that the model generalizes well and does not significantly overfit.

    - Minor fluctuations in training accuracy can be attributed to data augmentation and batch variations.
- **Loss Trend**:
    - The loss decreases rapidly during the initial epochs, suggesting effective early learning.
    - Both training and validation loss stabilize after ~25 epochs, indicating the model has reached a convergence point.
    - The model appears to avoid overfitting due to effective use of regularization techniques such as **Dropout**, **Batch Normalization**, and **EarlyStopping**.

8.2 Confusion Matrix Analysis

True Class	Most Confused With
Angry	Sad, Neutral
Disgust	Angry
Fear	Angry, Sad, Surprise
Happy	Mostly correctly classified
Sad	Angry, Neutral
Surprise	Fear, Happy
Neutral	Sad, Angry

8.3 Observations:

- **Happy** class has the highest correct predictions (1546/1774), suggesting that its facial features are distinct and easier for the CNN to recognize.
- **Disgust** and **Fear** are the most challenging classes, often misclassified as **Angry** or **Sad**, likely due to subtle differences in facial expressions.
- **Surprise** and **Neutral** show moderate classification success but also have confusion with classes like **Fear** and **Sad**.

8.4 Model and Optimization Insights

Hyperparameters:

- Optimizer: Adam with LR=0.0005 ensures smooth convergence.
- Loss Function: Categorical Crossentropy, suitable for multi-class classification.
- Epochs: 40 with EarlyStopping ensures training efficiency and prevents overfitting.
- Batch Size: 64 offers a balance between computation and learning stability.

Callbacks:

- **EarlyStopping** avoids overfitting by halting training when validation loss plateaus.
- **ReduceLROnPlateau** dynamically adjusts the learning rate to help the model converge better.

8.5 Model Strengths and Challenges

Strengths

- Excellent performance in recognizing well-defined emotions like **Happy** and **Neutral**.
- Effective use of augmentation and regularization helped improve generalization.
- Stable convergence and minimal overfitting.

Challenges

- Confusion among emotions with similar facial features like **Fear**, **Disgust**, and **Sad**.
- Imbalance in dataset (e.g., fewer **Disgust** samples) affected model learning.
- Lower recall in certain classes indicates missed detections even when predictions are precise.

9.Conclusion

The CNN-based facial emotion recognition model achieved a validation accuracy of approximately **65%**, showing strong performance in classifying emotions like **Happy** and **Neutral**, which have distinct facial features. However, it struggled with emotions such as **Disgust**, **Fear**, and **Surprise**, primarily due to inter-class similarities and class imbalance. The training and validation curves indicate smooth convergence with minimal overfitting, thanks to techniques like **Dropout**, **EarlyStopping**, and **learning rate scheduling**. The confusion matrix further highlights the model's tendency to misclassify similar emotions. Overall, while the model demonstrates good generalization, performance can be further improved by augmenting underrepresented classes and refining the network with attention mechanisms or advanced architectures.

10. Appendix

10.1 code:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (Conv2D, MaxPooling2D, Dense, Dropout, BatchNormalization,
                                     GlobalAveragePooling2D)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
train_dir = 'F:/Face-Emotion-recognition/data/train'
test_dir = 'F:/Face-Emotion-recognition/data/test'
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
img_size = 48
batch_size = 64
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=25,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_size, img_size),
    color_mode="grayscale",
    batch_size=batch_size,
    class_mode="categorical",
    shuffle=True
)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_size, img_size),
    color_mode="grayscale",
    batch_size=batch_size,
    class_mode="categorical",
    shuffle=False
)
def create_improved_model():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(img_size, img_size,
1)))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(GlobalAveragePooling2D())
    model.add(Dense(256, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.4))
    model.add(Dense(7, activation='softmax'))
    return model
model = create_improved_model()
model.compile(optimizer=Adam(learning_rate=0.0005),                    loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
early_stopping    =    EarlyStopping(monitor='val_loss',    patience=10,    restore_best_weights=True,
verbose=1)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=4, min_lr=1e-6, verbose=1)
steps_per_epoch = train_generator.samples // batch_size
validation_steps = test_generator.samples // batch_size

history = model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=40,
    validation_data=test_generator,
    validation_steps=validation_steps,
    callbacks=[early_stopping, reduce_lr]
)
def plot_history(history):
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Val')
    plt.title('Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Val')
    plt.title('Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.tight_layout()
    plt.show()
plot_history(history)
loss, acc = model.evaluate(test_generator)
print(f'Test Accuracy: {acc * 100:.2f}%')
y_pred = model.predict(test_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_generator.classes
print("Classification Report:")
print(classification_report(y_true, y_pred_classes, target_names=emotion_labels))
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=emotion_labels,
yticklabels=emotion_labels)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
model.save('improved_emotion_model.h5')
```