```
!unzip -q /content/fer2013_images.zip -d /content/
!ls /content
```

```
fer2013_images  fer2013_images.zip  sample_data
```

```
# ============================================================
# Nathalia Silva
# ============================================================
# Facial Emotion Recognition using CNNs (Images Version)
# Dataset format: train/ and test/ folders with subfolders per class
# ============================================================

# ---- 1. Imports ----
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 import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator

print("TensorFlow version:", tf.__version__)

# ---- 2. Basic Configuration ----
SEED = 42
np.random.seed(SEED)
tf.random.set_seed(SEED)

IMG_SIZE = (48, 48)
BATCH_SIZE = 64
EPOCHS = 35

BASE_DIR = "/content/fer2013_images"   # contains train/ and test/
train_dir = os.path.join(BASE_DIR, "train")
test_dir  = os.path.join(BASE_DIR, "test")

print("Train directory:", train_dir)
print("Test directory :", test_dir)
print("Train subdirectories:", os.listdir(train_dir))

# ---- 3. Data Generators ----

# Augmentation + automatic validation split (20% of training set)
train_datagen = ImageDataGenerator(
    rescale=1./255,
```

```
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    validation_split=0.2
)

# No augmentation for test set
test_datagen = ImageDataGenerator(rescale=1./255)

# Train generator
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    color_mode="grayscale",
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="training",
    shuffle=True,
    seed=SEED
)

# Validation generator
val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    color_mode="grayscale",
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    subset="validation",
    shuffle=False,
    seed=SEED
)

# Test generator
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    color_mode="grayscale",
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    shuffle=False
)

num_classes = train_generator.num_classes
class_indices = train_generator.class_indices
idx_to_class = {v: k for k, v in class_indices.items()}
print("Class indices:", class_indices)
```

```
# ---- 4. Build CNN Model ----

def build_cnn_model(input_shape=(48, 48, 1), num_classes=7):
    model = models.Sequential(name="FER2013_Images_CNN")

    # Block 1
    model.add(layers.Conv2D(32, (3, 3), activation="relu", padding="same",
                            input_shape=input_shape))
    model.add(layers.Conv2D(32, (3, 3), activation="relu", padding="same"))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(0.25))

    # Block 2
    model.add(layers.Conv2D(64, (3, 3), activation="relu", padding="same"))
    model.add(layers.Conv2D(64, (3, 3), activation="relu", padding="same"))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(0.25))

    # Block 3
    model.add(layers.Conv2D(128, (3, 3), activation="relu", padding="same"))
    model.add(layers.Conv2D(128, (3, 3), activation="relu", padding="same"))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(0.3))

    # Classifier
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation="relu"))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(num_classes, activation="softmax"))

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model

input_shape = (IMG_SIZE[0], IMG_SIZE[1], 1)
model = build_cnn_model(input_shape=input_shape, num_classes=num_classes)
model.summary()

# ---- 5. Callbacks ----
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

checkpoint_path = "/content/best_fer2013_images_cnn.keras"

callbacks = [
    EarlyStopping(
        monitor="val_accuracy",
        patience=8,
        restore_best_weights=True
```

```
        ),
        ReduceLROnPlateau(
            monitor="val_loss",
            factor=0.5,
            patience=4,
            min_lr=1e-6,
            verbose=1
        ),
        ModelCheckpoint(
            filepath=checkpoint_path,
            monitor="val_accuracy",
            save_best_only=True,
            verbose=1
        )
    ]

    # ---- 6. Train Model ----
    history = model.fit(
        train_generator,
        epochs=EPOCHS,
        validation_data=val_generator,
        callbacks=callbacks
    )

    # ---- 7. Plot Training History ----
    def plot_training_history(history):
        hist = history.history

        plt.figure(figsize=(14, 5))

        # Accuracy
        plt.subplot(1, 2, 1)
        plt.plot(hist["accuracy"], label="Train")
        plt.plot(hist["val_accuracy"], label="Validation")
        plt.title("Training and Validation Accuracy")
        plt.xlabel("Epoch")
        plt.ylabel("Accuracy")
        plt.legend()
        plt.grid(True)

        # Loss
        plt.subplot(1, 2, 2)
        plt.plot(hist["loss"], label="Train")
        plt.plot(hist["val_loss"], label="Validation")
        plt.title("Training and Validation Loss")
        plt.xlabel("Epoch")
        plt.ylabel("Loss")
        plt.legend()
        plt.grid(True)
```

```
        plt.tight_layout()
        plt.show()

plot_training_history(history)

# ---- 8. Evaluate on Test Set ----
test_loss, test_acc = model.evaluate(test_generator, verbose=1)
print("\nTest Accuracy: {:.2f}%".format(test_acc * 100))

# ---- 9. Confusion Matrix and Classification Report ----
y_prob = model.predict(test_generator)
y_pred = np.argmax(y_prob, axis=1)
y_true = test_generator.classes

target_names = [idx_to_class[i] for i in range(num_classes)]

print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=target_names))

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d",
            xticklabels=target_names,
            yticklabels=target_names)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix - CNN Emotion Classification")
plt.tight_layout()
plt.show()

# ---- 10. Save Final Model ----
final_model_path = "/content/fer2013_images_cnn_final.keras"
model.save(final_model_path)
print("Final model saved at:", final_model_path)
print("Best model saved at:", checkpoint_path)
```

```
TensorFlow version: 2.19.0
Train directory: /content/fer2013_images/train
Test directory : /content/fer2013_images/test
Train subdirectories: ['sad', 'surprise', 'disgust', 'angry', 'neutral', 'happy', 'fear']
Found 22968 images belonging to 7 classes.
Found 5741 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
Class indices: {'angry': 0, 'disgust': 1, 'fear': 2, 'happy': 3, 'neutral': 4, 'sad': 5, 'surprise': 6}
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argume
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "FER2013_Images_CNN"
```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 48, 48, 32)	320
conv2d_7 (Conv2D)	(None, 48, 48, 32)	9,248
max_pooling2d_3 (MaxPooling2D)	(None, 24, 24, 32)	0
dropout_4 (Dropout)	(None, 24, 24, 32)	0
conv2d_8 (Conv2D)	(None, 24, 24, 64)	18,496
conv2d_9 (Conv2D)	(None, 24, 24, 64)	36,928
max_pooling2d_4 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_5 (Dropout)	(None, 12, 12, 64)	0
conv2d_10 (Conv2D)	(None, 12, 12, 128)	73,856
conv2d_11 (Conv2D)	(None, 12, 12, 128)	147,584
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_6 (Dropout)	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 256)	1,179,904
dropout_7 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 7)	1,799

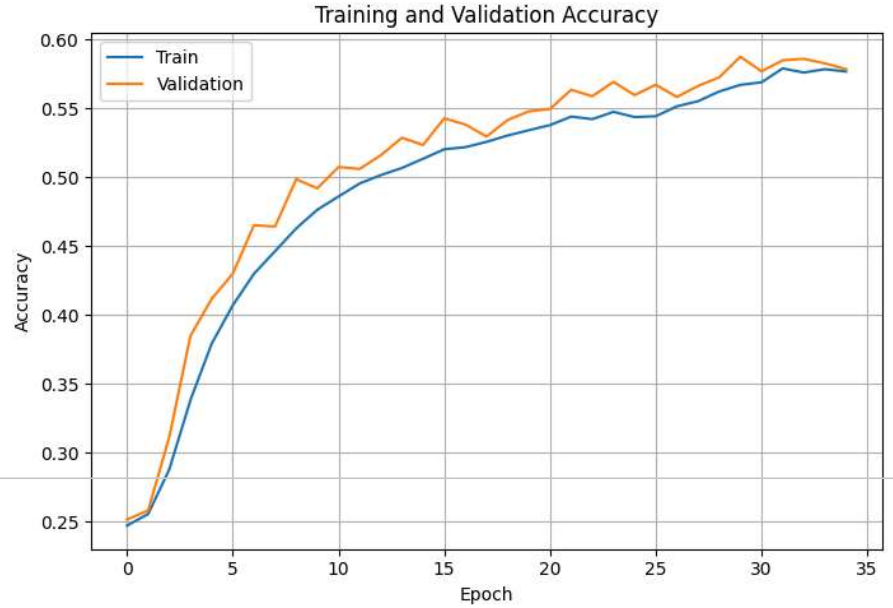
```
 Total params: 1,468,135 (5.60 MB)
 Trainable params: 1,468,135 (5.60 MB)
 Non-trainable params: 0 (0.00 B)
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `
  self._warn_if_super_not_called()
Epoch 1/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.2373 - loss: 1.8391
Epoch 1: val_accuracy improved from -inf to 0.25135, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 35s 77ms/step - accuracy: 0.2374 - loss: 1.8390 - val_accuracy: 0.2513 - val_loss: 1.8014 - learning_rate: 0.0010
Epoch 2/35
```

```
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.2523 - loss: 1.7932
Epoch 2: val_accuracy improved from 0.25135 to 0.25797, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.2523 - loss: 1.7931 - val_accuracy: 0.2580 - val_loss: 1.7688 - learning_rate: 0.0010
Epoch 3/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.2787 - loss: 1.7530
Epoch 3: val_accuracy improved from 0.25797 to 0.31144, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.2787 - loss: 1.7530 - val_accuracy: 0.3114 - val_loss: 1.7059 - learning_rate: 0.0010
Epoch 4/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.3194 - loss: 1.6916
Epoch 4: val_accuracy improved from 0.31144 to 0.38478, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 21s 59ms/step - accuracy: 0.3195 - loss: 1.6914 - val_accuracy: 0.3848 - val_loss: 1.5822 - learning_rate: 0.0010
Epoch 5/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.3767 - loss: 1.5907
Epoch 5: val_accuracy improved from 0.38478 to 0.41160, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.3767 - loss: 1.5907 - val_accuracy: 0.4116 - val_loss: 1.5174 - learning_rate: 0.0010
Epoch 6/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.4015 - loss: 1.5263
Epoch 6: val_accuracy improved from 0.41160 to 0.42989, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.4015 - loss: 1.5263 - val_accuracy: 0.4299 - val_loss: 1.4586 - learning_rate: 0.0010
Epoch 7/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.4202 - loss: 1.4805
Epoch 7: val_accuracy improved from 0.42989 to 0.46508, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.4203 - loss: 1.4805 - val_accuracy: 0.4651 - val_loss: 1.3964 - learning_rate: 0.0010
Epoch 8/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.4453 - loss: 1.4260
Epoch 8: val_accuracy did not improve from 0.46508
359/359 ━━━━━━━━━━━━━━━━━━━━ 21s 60ms/step - accuracy: 0.4453 - loss: 1.4260 - val_accuracy: 0.4642 - val_loss: 1.3809 - learning_rate: 0.0010
Epoch 9/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.4562 - loss: 1.3974
Epoch 9: val_accuracy improved from 0.46508 to 0.49852, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.4562 - loss: 1.3974 - val_accuracy: 0.4985 - val_loss: 1.3161 - learning_rate: 0.0010
Epoch 10/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.4721 - loss: 1.3682
Epoch 10: val_accuracy did not improve from 0.49852
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 60ms/step - accuracy: 0.4721 - loss: 1.3682 - val_accuracy: 0.4919 - val_loss: 1.3185 - learning_rate: 0.0010
Epoch 11/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.4880 - loss: 1.3348
Epoch 11: val_accuracy improved from 0.49852 to 0.50740, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.4880 - loss: 1.3348 - val_accuracy: 0.5074 - val_loss: 1.2816 - learning_rate: 0.0010
Epoch 12/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.4894 - loss: 1.3320
Epoch 12: val_accuracy did not improve from 0.50740
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.4894 - loss: 1.3320 - val_accuracy: 0.5060 - val_loss: 1.2694 - learning_rate: 0.0010
Epoch 13/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.5011 - loss: 1.3058
Epoch 13: val_accuracy improved from 0.50740 to 0.51594, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 21s 59ms/step - accuracy: 0.5011 - loss: 1.3058 - val_accuracy: 0.5159 - val_loss: 1.2354 - learning_rate: 0.0010
Epoch 14/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5027 - loss: 1.2975
Epoch 14: val_accuracy improved from 0.51594 to 0.52865, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.5027 - loss: 1.2974 - val_accuracy: 0.5287 - val_loss: 1.2263 - learning_rate: 0.0010
Epoch 15/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5150 - loss: 1.2695
Epoch 15: val_accuracy did not improve from 0.52865
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 61ms/step - accuracy: 0.5150 - loss: 1.2695 - val_accuracy: 0.5234 - val_loss: 1.2492 - learning_rate: 0.0010
```

```
Epoch 16/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5204 - loss: 1.2664
Epoch 16: val_accuracy improved from 0.52865 to 0.54276, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5204 - loss: 1.2664 - val_accuracy: 0.5428 - val_loss: 1.2009 - learning_rate: 0.0010
Epoch 17/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.5204 - loss: 1.2511
Epoch 17: val_accuracy did not improve from 0.54276
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5204 - loss: 1.2511 - val_accuracy: 0.5382 - val_loss: 1.1933 - learning_rate: 0.0010
Epoch 18/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.5228 - loss: 1.2524
Epoch 18: val_accuracy did not improve from 0.54276
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 60ms/step - accuracy: 0.5228 - loss: 1.2524 - val_accuracy: 0.5295 - val_loss: 1.2189 - learning_rate: 0.0010
Epoch 19/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5268 - loss: 1.2368
Epoch 19: val_accuracy did not improve from 0.54276
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5268 - loss: 1.2368 - val_accuracy: 0.5415 - val_loss: 1.2088 - learning_rate: 0.0010
Epoch 20/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5334 - loss: 1.2235
Epoch 20: val_accuracy improved from 0.54276 to 0.54781, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5334 - loss: 1.2235 - val_accuracy: 0.5478 - val_loss: 1.1833 - learning_rate: 0.0010
Epoch 21/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5422 - loss: 1.2149
Epoch 21: val_accuracy improved from 0.54781 to 0.54956, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 63ms/step - accuracy: 0.5422 - loss: 1.2149 - val_accuracy: 0.5496 - val_loss: 1.1751 - learning_rate: 0.0010
Epoch 22/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5460 - loss: 1.2040
Epoch 22: val_accuracy improved from 0.54956 to 0.56349, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5460 - loss: 1.2040 - val_accuracy: 0.5635 - val_loss: 1.1557 - learning_rate: 0.0010
Epoch 23/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5462 - loss: 1.1964
Epoch 23: val_accuracy did not improve from 0.56349
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 64ms/step - accuracy: 0.5462 - loss: 1.1964 - val_accuracy: 0.5588 - val_loss: 1.1538 - learning_rate: 0.0010
Epoch 24/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5500 - loss: 1.1996
Epoch 24: val_accuracy improved from 0.56349 to 0.56924, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5500 - loss: 1.1997 - val_accuracy: 0.5692 - val_loss: 1.1314 - learning_rate: 0.0010
Epoch 25/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5451 - loss: 1.1916
Epoch 25: val_accuracy did not improve from 0.56924
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5451 - loss: 1.1917 - val_accuracy: 0.5597 - val_loss: 1.1755 - learning_rate: 0.0010
Epoch 26/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.5398 - loss: 1.2050
Epoch 26: val_accuracy did not improve from 0.56924
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5398 - loss: 1.2050 - val_accuracy: 0.5671 - val_loss: 1.1462 - learning_rate: 0.0010
Epoch 27/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.5488 - loss: 1.1843
Epoch 27: val_accuracy did not improve from 0.56924
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 64ms/step - accuracy: 0.5488 - loss: 1.1843 - val_accuracy: 0.5583 - val_loss: 1.1595 - learning_rate: 0.0010
Epoch 28/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5614 - loss: 1.1686
Epoch 28: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 28: val_accuracy did not improve from 0.56924
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5614 - loss: 1.1687 - val_accuracy: 0.5663 - val_loss: 1.1317 - learning_rate: 0.0010
Epoch 29/35
```

```
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - accuracy: 0.5583 - loss: 1.1582
Epoch 29: val_accuracy improved from 0.56924 to 0.57255, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5584 - loss: 1.1582 - val_accuracy: 0.5725 - val_loss: 1.1208 - learning_rate: 5.0000e-04
Epoch 30/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5673 - loss: 1.1375
Epoch 30: val_accuracy improved from 0.57255 to 0.58753, saving model to /content/best_fer2013_images_cnn.keras
359/359 ━━━━━━━━━━━━━━━━━━━━ 22s 62ms/step - accuracy: 0.5673 - loss: 1.1376 - val_accuracy: 0.5875 - val_loss: 1.0983 - learning_rate: 5.0000e-04
Epoch 31/35
358/359 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.5768 - loss: 1.1209
Epoch 31: val_accuracy did not improve from 0.58753
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 64ms/step - accuracy: 0.5768 - loss: 1.1210 - val_accuracy: 0.5769 - val_loss: 1.1203 - learning_rate: 5.0000e-04
Epoch 32/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.5822 - loss: 1.1165
Epoch 32: val_accuracy did not improve from 0.58753
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5822 - loss: 1.1166 - val_accuracy: 0.5849 - val_loss: 1.1035 - learning_rate: 5.0000e-04
Epoch 33/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5755 - loss: 1.1280
Epoch 33: val_accuracy did not improve from 0.58753
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5755 - loss: 1.1280 - val_accuracy: 0.5860 - val_loss: 1.0962 - learning_rate: 5.0000e-04
Epoch 34/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step - accuracy: 0.5752 - loss: 1.1379
Epoch 34: val_accuracy did not improve from 0.58753
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 65ms/step - accuracy: 0.5752 - loss: 1.1378 - val_accuracy: 0.5827 - val_loss: 1.1030 - learning_rate: 5.0000e-04
Epoch 35/35
359/359 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.5747 - loss: 1.1207
Epoch 35: val_accuracy did not improve from 0.58753
359/359 ━━━━━━━━━━━━━━━━━━━━ 23s 63ms/step - accuracy: 0.5747 - loss: 1.1207 - val_accuracy: 0.5785 - val_loss: 1.1022 - learning_rate: 5.0000e-04
```




```
113/113 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - accuracy: 0.5388 - loss: 1.1570

Test Accuracy: 59.53%
113/113 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step
```

```
Classification Report:
              precision    recall  f1-score   support

       angry       0.51      0.53      0.52       958
     disgust       0.55      0.14      0.23       111
        fear       0.47      0.23      0.31      1024
       happy       0.78      0.87      0.83      1774
     neutral       0.49      0.64      0.56      1233
         sad       0.46      0.43      0.44      1247
    surprise       0.71      0.77      0.74       831

    accuracy                           0.60      7178
   macro avg       0.57      0.52      0.52      7178
weighted avg       0.58      0.60      0.58      7178
```