```
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Load the dataset
data = pd.read_csv('fer2013.csv')  # Update with the path to your dataset

# Preprocess the data
X = []
y = []

for index, row in data.iterrows():
    img = np.array(row['pixels'].split(), dtype='float32').reshape(48, 48) / 255.0  # Normalize
    X.append(img)
    y.append(row['emotion'])

X = np.array(X)
y = to_categorical(np.array(y))  # One-hot encode the labels

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Reshape for CNN input
X_train = X_train.reshape(-1, 48, 48, 1)
X_test = X_test.reshape(-1, 48, 48, 1)

# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(7, activation='softmax')  # 7 classes for emotions
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=300, batch_size=64, validation_data=(X_test, y_test))

# Save the model
model.save('emotion_recognition_model.h5')
```

```
Epoch 73/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 142s 192ms/step - accuracy: 0.8467 - loss: 0.3562 - val_accuracy: 0.5460 - val_loss: 3.0532
Epoch 74/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 84s 187ms/step - accuracy: 0.8531 - loss: 0.3408 - val_accuracy: 0.5454 - val_loss: 3.2623
Epoch 75/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 89s 198ms/step - accuracy: 0.8516 - loss: 0.3467 - val_accuracy: 0.5397 - val_loss: 3.2191
Epoch 76/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 84s 187ms/step - accuracy: 0.8543 - loss: 0.3444 - val_accuracy: 0.5435 - val_loss: 3.1647
Epoch 77/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 90s 200ms/step - accuracy: 0.8571 - loss: 0.3393 - val_accuracy: 0.5411 - val_loss: 3.3970
Epoch 78/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 137s 189ms/step - accuracy: 0.8543 - loss: 0.3352 - val_accuracy: 0.5400 - val_loss: 3.1963
Epoch 79/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 142s 189ms/step - accuracy: 0.8591 - loss: 0.3325 - val_accuracy: 0.5378 - val_loss: 3.2316
Epoch 80/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 143s 191ms/step - accuracy: 0.8601 - loss: 0.3291 - val_accuracy: 0.5366 - val_loss: 3.2366
Epoch 81/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 143s 194ms/step - accuracy: 0.8580 - loss: 0.3347 - val_accuracy: 0.5439 - val_loss: 3.3703
Epoch 82/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 146s 202ms/step - accuracy: 0.8581 - loss: 0.3350 - val_accuracy: 0.5418 - val_loss: 3.2906
Epoch 83/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 136s 189ms/step - accuracy: 0.8582 - loss: 0.3246 - val_accuracy: 0.5450 - val_loss: 3.3490
Epoch 84/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 142s 189ms/step - accuracy: 0.8581 - loss: 0.3369 - val_accuracy: 0.5440 - val_loss: 3.3729
Epoch 85/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 87s 194ms/step - accuracy: 0.8611 - loss: 0.3258 - val_accuracy: 0.5421 - val_loss: 3.2841
Epoch 86/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 143s 196ms/step - accuracy: 0.8586 - loss: 0.3355 - val_accuracy: 0.5428 - val_loss: 3.3555
Epoch 87/300
449/449 ━━━━━━━━━━━━━━━━━━━━ 139s 188ms/step - accuracy: 0.8606 - loss: 0.3295 - val_accuracy: 0.5396 - val_loss: 3.4658
Epoch 88/300
 16/449 ━━━━━━━━━━━━━━━━━━━━ 1:46 246ms/step - accuracy: 0.8726 - loss: 0.2733
```

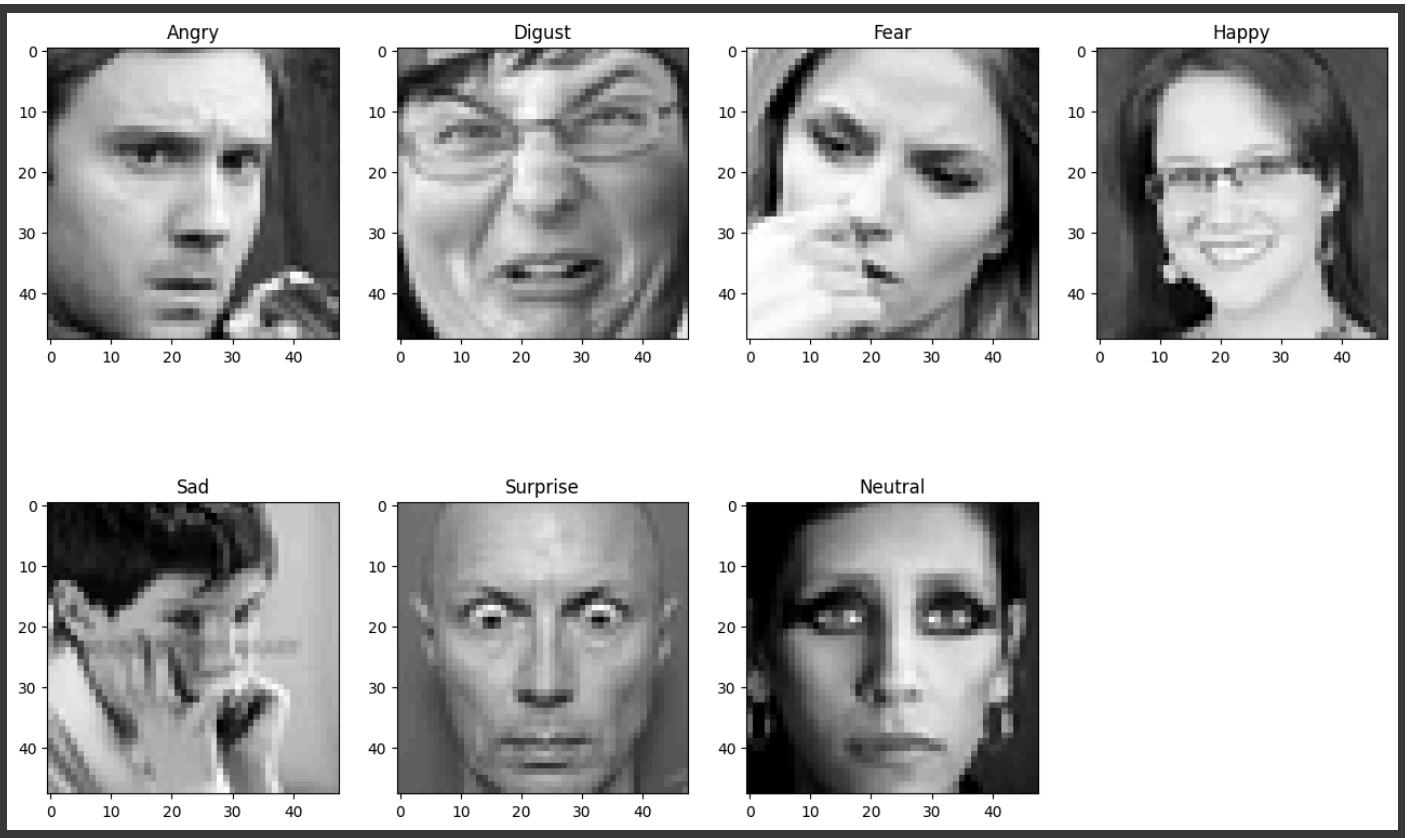
```
!unzip fer2013.csv.zip
```

```
Archive:  fer2013.csv.zip
  inflating: fer2013.csv
```

```
df = pd.read_csv('fer2013.csv')
emotions = {0: 'Angry', 1: 'Digust', 2: 'Fear', 3: 'Happy', 4: 'Sad', 5: 'Surprise', 6: 'Neutral'}
emotion_counts = df['emotion'].value_counts(sort=False).reset_index()
emotion_counts.columns = ['emotion', 'number']
emotion_counts['emotion'] = emotion_counts['emotion'].map(emotions)
emotion_counts
```

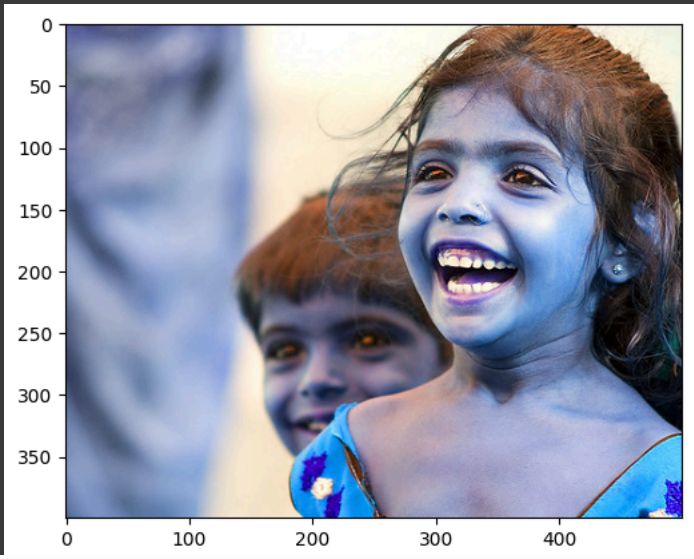
	emotion	number
0	Angry	4953
1	Fear	5121
2	Sad	6077
3	Neutral	6198
4	Happy	8989
5	Surprise	4002
6	Digust	547

```
def pictures(row):
    pixels, emotion = row['pixels'], emotions[row['emotion']]
    img = np.array(pixels.split(), dtype=int)
    img = img.reshape(48,48)
    return img, emotion

plt.figure(0, figsize=(16,10))
for i in range(1,9):
    if len(df[df['emotion'] == i-1]) == 0:
        continue

    face = df[df['emotion'] == i-1].iloc[0]
    img, emotion = pictures(face)
    plt.subplot(2,4,i)
    plt.imshow(img, cmap='gray')
    plt.title(emotion)

plt.show()
```




```
import cv2
import numpy as np
from tensorflow.keras.models import load_model

# Load the trained model from .h5 file
model = load_model('/content/emotion_recognition_model.h5')  # Replace with your .h5 file path

# Define the emotions corresponding to the model's output
emotions = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

# Function to preprocess the image
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    plt.imshow(img, cmap='gray')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    img = cv2.resize(img, (48, 48))  # Resize to the expected input size
    img = img / 255.0  # Normalize the image
    img = img.reshape(1, 48, 48, 1)  # Add batch and channel dimensions
    return img

# Function to predict emotion from an image
def predict_emotion(image_path):
    img = preprocess_image(image_path)
    prediction = model.predict(img)
    predicted_index = np.argmax(prediction)  # Get the index of the highest probability
    emotion_name = emotions[predicted_index]  # Map the index to the emotion name
    return emotion_name

# Example usage
if __name__ == "__main__":
    image_path = 'img.jpg'  # Replace with your image path
    predicted_emotion = predict_emotion(image_path)
    print(f'Predicted Emotion: {predicted_emotion}')
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
Predicted Emotion: Fear
```




```
from google.colab.patches import cv2_imshow
```

```
Start coding or generate with AI.
```