Group 11

1. Youssef Ossama Foaud
2. Zeyad Mohamed Asran
3. Ahmed Mohamed Adel
4. Mohamed Ossama Mohamed

Project Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
import cv2
import random
import tensorflow as tf
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras_preprocessing.image import load_img
from keras import layers
from keras.metrics import categorical_crossentropy
from keras.models import Model, Sequential, load_model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout, MaxPooling2D,
Flatten, Conv2D
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.initializers import glorot_uniform
```

```
import splitfolders
input_folder = r'Z:\projects\amm\dataset\CK+48'
output_folder = r'Z:\projects\amm\dataset'
splitfolders.ratio(input_folder, output_folder, seed=42, ratio=(0.8, 0.1, 0.1))
```

```
target_size=(48,48)
np.random.seed(42)
tf.random.set_seed(42)
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    horizontal_flip=True)
```

```
train_generator = train_datagen.flow_from_directory(
    'Z:\\projects\\amm\\dataset\\train',
    target_size=target_size,
    color_mode='rgb',
    batch_size=64,
    class_mode='categorical')

validation_generator = train_datagen.flow_from_directory(
    'Z:\\projects\\amm\\dataset\\val',
    target_size=target_size,
    color_mode='rgb',
    batch_size=64,
    class_mode='categorical')

datagen = ImageDataGenerator()
test_generator = datagen.flow_from_directory(
    'Z:\\projects\\amm\\dataset\\test',
    target_size=target_size,
    color_mode='rgb',
    batch_size=64,
    class_mode='categorical')
```

```
train_generator.image_shape
train_generator.class_indices
```

```
model = tf.keras.Sequential([
   tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,3)),
   tf.keras.layers.MaxPool2D(pool_size=(2, 2)),
   tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
   tf.keras.layers.MaxPool2D(pool_size=(2, 2)),
   tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
   tf.keras.layers.MaxPool2D(pool_size=(2, 2)),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(128, activation='relu'),
   tf.keras.layers.Dropout(rate=0.4),
   tf.keras.layers.Dense(7, activation='softmax')
])
```

```
optimizer = tf.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
model.summary()
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=8, verbose=1)
history = model.fit(train_generator, epochs=80,
validation_data=validation_generator, callbacks=[early_stopping])
```

```
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(accuracy) + 1)
#Train and validation accuracy
plt.plot(epochs, accuracy, 'b', label='Training accurarcy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure() # to plot new figure
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test Accuracy: {test_acc}')
```

```
model.save("modelfinalll")
```

```
label = ['angry','neutral','disgust','fear','happy','sad','surprise']
def predict_emotion(image_path, model, label):
    img = tf.keras.preprocessing.image.load_img(image_path, target_size=(48, 48))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)
    img_array = img_array / 255.0
    predictions = model.predict(img_array)
    predicted_class_index = np.argmax(predictions)
    emotion_labels = label
    predicted_emotion = emotion_labels[predicted_class_index]
    for label, score in zip(emotion_labels, predictions[0]):
        print(f"{label.capitalize()}: {score * 100:.2f}%")
    plt.imshow(img, interpolation='nearest')
    plt.title(f'Predicted Emotion: {predicted_emotion}')
    plt.axis('off')
    plt.show()

image_paths_to_predict = [
    r'Z:\projects\amm\dataset\test\anger\S014_003_00000028.png',
    r'Z:\projects\amm\dataset\test\fear\S032_004_00000012.png',
    r'Z:\projects\amm\dataset\test\surprise\S014_001_00000027.png',
    r'Z:\projects\amm\dataset\test\sadness\S093_001_00000019.png'
]

for image_path in image_paths_to_predict:
    predict_emotion(image_path, model, label)
```