## Manahil Fatima Anwar

## 20K-0134

## BAI-7A

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
!unzip 'archive (1).zip'
```

```
      inflating: train/surprise/Training_88282683.jpg
      inflating: train/surprise/Training_88289855.jpg
      inflating: train/surprise/Training_8829877.jpg
      inflating: train/surprise/Training_88312551.jpg
      inflating: train/surprise/Training_88374639.jpg
      inflating: train/surprise/Training_88395708.jpg
      inflating: train/surprise/Training_88480489.jpg
      inflating: train/surprise/Training_88490087.jpg
      inflating: train/surprise/Training_88503708.jpg
      inflating: train/surprise/Training_88523782.jpg
      inflating: train/surprise/Training_88548757.jpg
      inflating: train/surprise/Training_88570558.jpg
      inflating: train/surprise/Training_88690760.jpg
      inflating: train/surprise/Training_88703294.jpg
      inflating: train/surprise/Training_88784237.jpg
      inflating: train/surprise/Training_88818523.jpg
      inflating: train/surprise/Training_88876320.jpg
      inflating: train/surprise/Training_88900715.jpg
      inflating: train/surprise/Training_88916108.jpg
      inflating: train/surprise/Training_88959397.jpg
      inflating: train/surprise/Training_89006794.jpg
      inflating: train/surprise/Training_8905218.jpg
      inflating: train/surprise/Training_89065069.jpg
      inflating: train/surprise/Training_89087843.jpg
      inflating: train/surprise/Training_89097672.jpg
      inflating: train/surprise/Training_89107006.jpg
      inflating: train/surprise/Training_89137795.jpg
      inflating: train/surprise/Training_89139325.jpg
      inflating: train/surprise/Training_89149496.jpg
      inflating: train/surprise/Training_89216287.jpg
      inflating: train/surprise/Training_89233153.jpg
      inflating: train/surprise/Training_89259456.jpg
      inflating: train/surprise/Training_89263813.jpg
      inflating: train/surprise/Training_89274966.jpg
      inflating: train/surprise/Training_89305564.jpg
      inflating: train/surprise/Training_8932145.jpg
      inflating: train/surprise/Training_89353263.jpg
      inflating: train/surprise/Training_89378565.jpg
      inflating: train/surprise/Training_89380464.jpg
      inflating: train/surprise/Training_89421614.jpg
      inflating: train/surprise/Training_89423258.jpg
      inflating: train/surprise/Training_89470182.jpg
      inflating: train/surprise/Training_89511690.jpg
      inflating: train/surprise/Training_89517922.jpg
      inflating: train/surprise/Training_89521599.jpg
      inflating: train/surprise/Training_89549550.jpg
      inflating: train/surprise/Training_89567115.jpg
      inflating: train/surprise/Training_89587019.jpg
      inflating: train/surprise/Training_89600129.jpg
      inflating: train/surprise/Training_89611881.jpg
      inflating: train/surprise/Training_8961382.jpg
      inflating: train/surprise/Training_89621882.jpg
      inflating: train/surprise/Training_89627155.jpg
      inflating: train/surprise/Training_89692529.jpg
      inflating: train/surprise/Training_89739298.jpg
      inflating: train/surprise/Training_89800588.jpg
      inflating: train/surprise/Training_89837846.jpg
      inflating: train/surprise/Training_89923105.jpg
      inflating: train/surprise/Training_89930272.jpg
```

```
# Define constants
img_height, img_width = 224, 224
num_classes = 7
batch_size = 32

# Define data paths
train_data_dir = 'train'
test_data_dir = 'test'

# Data preprocessing and augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

# Data generators
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

```
# Load pre-trained ResNet model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

# Freeze the layers of the pre-trained ResNet
for layer in base_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_nc
94765736/94765736 [==============================] - 0s 0us/step
```

```
# Build your classification model on top of the pre-trained ResNet
model = models.Sequential()
model.add(base_model)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(7, activation='softmax'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))
```

```
# Compile the model
from tensorflow.keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model
epochs = 10  # Adjust the number of epochs as needed
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=test_generator
)
```

```
Epoch 1/10
898/898 [==============================] - 395s 436ms/step - loss: 1.9033 - accuracy: 0.2096 - val_loss: 1.8772 - val_accuracy: 0.2438
Epoch 2/10
898/898 [==============================] - 404s 450ms/step - loss: 1.8804 - accuracy: 0.2187 - val_loss: 1.8563 - val_accuracy: 0.2471
Epoch 3/10
898/898 [==============================] - 392s 437ms/step - loss: 1.8617 - accuracy: 0.2276 - val_loss: 1.8438 - val_accuracy: 0.2471
Epoch 4/10
```

```
898/898 [==============================] - 389s 433ms/step - loss: 1.8521 - accuracy: 0.2314 - val_loss: 1.8362 - val_accuracy: 0.2471
Epoch 5/10
898/898 [==============================] - 388s 432ms/step - loss: 1.8446 - accuracy: 0.2326 - val_loss: 1.8293 - val_accuracy: 0.2471
Epoch 6/10
898/898 [==============================] - 384s 427ms/step - loss: 1.8385 - accuracy: 0.2463 - val_loss: 1.8252 - val_accuracy: 0.2471
Epoch 7/10
898/898 [==============================] - 407s 453ms/step - loss: 1.8346 - accuracy: 0.2513 - val_loss: 1.8230 - val_accuracy: 0.2471
Epoch 8/10
898/898 [==============================] - 400s 445ms/step - loss: 1.8314 - accuracy: 0.2513 - val_loss: 1.8202 - val_accuracy: 0.2471
Epoch 9/10
898/898 [==============================] - 387s 431ms/step - loss: 1.8276 - accuracy: 0.2513 - val_loss: 1.8181 - val_accuracy: 0.2471
Epoch 10/10
898/898 [==============================] - 386s 430ms/step - loss: 1.8257 - accuracy: 0.2513 - val_loss: 1.8181 - val_accuracy: 0.2471
```

```
# Evaluate the model on the test set
accuracy = model.evaluate(test_generator)[1]
print('Test Accuracy: {:.2%}'.format(accuracy))
```

```
225/225 [==============================] - 26s 113ms/step - loss: 1.8181 - accuracy: 0.2471
Test Accuracy: 24.71%
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load the image
img = mpimg.imread('happy_lady.jpg')

# Display the image
plt.imshow(img)
plt.axis('off')
plt.show()

# Make predictions on a single image or a batch of images
def predict_image(model, img_path):
    img = tf.keras.preprocessing.image.load_img(img_path, target_size=(img_height, img_width))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array /= 255.0  # Normalize pixel values to between 0 and 1

    # Get predictions for the image
    predictions = model.predict(img_array)

    # Get the predicted class
    predicted_class = np.argmax(predictions)

    return predicted_class

emotion_labels = ["Angry", "Disgust", "Fear", "Happy", "Sad", "Surprise", "Neutral"]

image_path = 'happy_lady.jpg'
predicted_class = predict_image(model, image_path)

# Display the predicted class
predicted_label = emotion_labels[predicted_class]
print('Predicted Emotion:', predicted_label)
```