```
import numpy as np
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remo
```

```
# Load the image using OpenCV
img = cv2.imread('/content/gdrive/MyDrive/Test_set/Ripness tommato/36.jpg')
# Check if the image was loaded successfully
if img is None:
    print("Image load failed!")
else:
    # Resize the image to the expected input size (e.g., 64x64)
    img = cv2.resize(img, (64, 64))

    # Normalize pixel values to the range [0, 1]
    img = img / 255.0

    # Create a Sequential model for ripeness classification
    classifier = Sequential()

    # Add Convolutional layers
    classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
    classifier.add(MaxPooling2D(pool_size=(2, 2)))
    classifier.add(Conv2D(32, (3, 3), activation='relu'))
    classifier.add(MaxPooling2D(pool_size=(2, 2)))

    # Flatten the feature maps
    classifier.add(Flatten())

    # Add Fully Connected layers
    classifier.add(Dense(units=128, activation='relu'))
    classifier.add(Dense(units=1, activation='sigmoid'))

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

    # Assuming you have a dataset with labeled tomato images
    # You should have two subdirectories, e.g., 'ripe' and 'unripe', each containing respective images
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

    training_set = train_datagen.flow_from_directory(
        '/content/gdrive/MyDrive/Training_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

    # Train the model
    classifier.fit_generator(
        training_set,
        steps_per_epoch=len(training_set),
        epochs=10)  # You may adjust the number of epochs

    # Make a prediction using the preprocessed image
    ripeness_prediction = classifier.predict(np.expand_dims(img, axis=0))[0][0]

    print("Ripeness Prediction:", ripeness_prediction)
```

```
Found 56 images belonging to 2 classes.
Epoch 1/10
<ipython-input-149-f722e1b4c330>:47: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future ve
  classifier.fit_generator(
2/2 [==============================] - 2s 109ms/step - loss: 0.6348 - accuracy: 0.4464
Epoch 2/10
2/2 [==============================] - 0s 82ms/step - loss: 0.3356 - accuracy: 0.9107
Epoch 3/10
2/2 [==============================] - 0s 131ms/step - loss: 0.0981 - accuracy: 1.0000
Epoch 4/10
2/2 [==============================] - 0s 81ms/step - loss: 0.0157 - accuracy: 1.0000
Epoch 5/10
2/2 [==============================] - 0s 79ms/step - loss: 0.0074 - accuracy: 1.0000
Epoch 6/10
2/2 [==============================] - 0s 62ms/step - loss: 0.0034 - accuracy: 1.0000
Epoch 7/10
```

```
2/2 [==============================] - 0s 125ms/step - loss: 3.8573e-04 - accuracy: 1.0000
Epoch 8/10
2/2 [==============================] - 0s 82ms/step - loss: 8.5182e-05 - accuracy: 1.0000
Epoch 9/10
2/2 [==============================] - 0s 73ms/step - loss: 3.2574e-05 - accuracy: 1.0000
Epoch 10/10
2/2 [==============================] - 0s 121ms/step - loss: 3.6366e-05 - accuracy: 1.0000
1/1 [==============================] - 0s 54ms/step
Ripeness Prediction: 0.9999485
```

```
# Assuming you have a test dataset similar to the training dataset
test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(
    '/content/gdrive/MyDrive/Test_set',
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary')

# Evaluate the model on the test set
test_accuracy = classifier.evaluate_generator(test_set, steps=len(test_set))
print("Test Accuracy:", test_accuracy[1])
```

```
Found 16 images belonging to 2 classes.
<ipython-input-150-bca56993a158>:11: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a futu
  test_accuracy = classifier.evaluate_generator(test_set, steps=len(test_set))
WARNING:tensorflow:6 out of the last 9 calls to <function Model.make_test_function.<locals>.test_function at 0x79b4c21ba
Test Accuracy: 1.0
```

```
import os

# Directory containing the random images
test_image_directory = '/content/gdrive/MyDrive/Data used for Test the model'

# Initialize a list to store predictions
predictions = []

# Iterate through the images in the directory
for filename in os.listdir(test_image_directory):
    if filename.endswith(".jpg"):
        image_path = os.path.join(test_image_directory, filename)

        # Preprocess the image
        preprocessed_image = preprocess_image(image_path)

        if preprocessed_image is not None:
            # Make a prediction using the preprocessed image
            ripeness_prediction = classifier.predict(np.expand_dims(preprocessed_image, axis=0))[0][0]
            predictions.append((filename, ripeness_prediction))
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
```

```
# Assuming you have loaded and preprocessed the random images and stored predictions in the 'predictions' list
# 'predictions' should contain tuples with filename and ripeness prediction

# Define ground truth labels (0 for unripe, 1 for ripe) based on your knowledge
ground_truth_labels = {
    '27.jpg': 1,
    '31.jpg': 1,
    '34.jpg': 1,
    '35.jpg': 1,
    'Test.jpg': 1,
    'Test1.jpg': 1,
    'Test2.jpg': 1,
    'Test3.jpg': 0,
    'Test14.jpg': 0
}

# Initialize variables to keep track of correct and total predictions
correct_predictions = 0
total_predictions = len(predictions)
```

```
# Compare model's predictions to ground truth labels
for filename, ripeness_prediction in predictions:
    # Get the ground truth label for this image
    ground_truth = ground_truth_labels.get(filename)

    if ground_truth is not None:
        # Convert the ripeness prediction to a binary label (0 or 1)
        predicted_label = 1 if ripeness_prediction >= 0.5 else 0
        print(ground_truth)
        # Check if the model's prediction matches the ground truth
        if predicted_label == ground_truth:
            correct_predictions += 1

# Calculate accuracy
accuracy = correct_predictions / total_predictions

# Print the accuracy

print("Accuracy on Random Images:", accuracy)
```

```
1
1
1
1
1
0
1
1
Accuracy on Random Images: 0.7777777777777778
```