In [ ]:

```
import tensorflow as tf
from tensorflow.keras import layers, models

train_dir = '/Train_Folder_out/'
validation_dir = '/Validation_Folder_out/'

img_size = (64, 64)
batch_size = 32
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=img_size,
    batch_size=batch_size,
)
validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    validation_dir,
    image_size=img_size,
    batch_size=batch_size,
)
model = models.Sequential([
    layers.Conv2D(32, (4, 4), activation='relu', input_shape=(64, 64, 3)),
    layers.AveragePooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(3, activation='softmax')
])
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset
)

test_loss, test_acc = model.evaluate(validation_dataset)
print('Test accuracy:', test_acc)
model.summary()
```

```
Found 1200 files belonging to 3 classes.
Found 487 files belonging to 3 classes.
Epoch 1/10
38/38 [==============================] - 5s 116ms/step - loss: 65.0976 - accuracy:
0.8200 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 2/10
38/38 [==============================] - 4s 112ms/step - loss: 1.6156e-04 - accura
cy: 1.0000 - val_loss: 3.7426e-06 - val_accuracy: 1.0000
Epoch 3/10
38/38 [==============================] - 4s 108ms/step - loss: 3.8107e-07 - accura
cy: 1.0000 - val_loss: 6.4930e-06 - val_accuracy: 1.0000
Epoch 4/10
38/38 [==============================] - 4s 115ms/step - loss: 5.4141e-08 - accura
cy: 1.0000 - val_loss: 5.0649e-06 - val_accuracy: 1.0000
Epoch 5/10
38/38 [==============================] - 4s 102ms/step - loss: 3.5564e-08 - accura
cy: 1.0000 - val_loss: 4.9085e-06 - val_accuracy: 1.0000
Epoch 6/10
38/38 [==============================] - 4s 116ms/step - loss: 2.8014e-08 - accura
cy: 1.0000 - val_loss: 5.3857e-06 - val_accuracy: 1.0000
Epoch 7/10
38/38 [==============================] - 5s 118ms/step - loss: 2.2252e-08 - accura
cy: 1.0000 - val_loss: 5.9006e-06 - val_accuracy: 1.0000
Epoch 8/10
38/38 [==============================] - 5s 124ms/step - loss: 1.7881e-08 - accura
cy: 1.0000 - val_loss: 6.0246e-06 - val_accuracy: 1.0000
Epoch 9/10
38/38 [==============================] - 5s 116ms/step - loss: 1.4901e-08 - accura
cy: 1.0000 - val_loss: 6.8982e-06 - val_accuracy: 1.0000
Epoch 10/10
38/38 [==============================] - 5s 130ms/step - loss: 1.2418e-08 - accura
cy: 1.0000 - val_loss: 7.4136e-06 - val_accuracy: 1.0000
16/16 [==============================] - 1s 28ms/step - loss: 7.4136e-06 - accurac
y: 1.0000
Test accuracy: 1.0
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 61, 61, 32)        1568

 average_pooling2d_12 (Avera  (None, 30, 30, 32)       0
 gePooling2D)

 conv2d_19 (Conv2D)          (None, 28, 28, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_6 (Flatten)         (None, 12544)             0

 dense_18 (Dense)            (None, 128)               1605760

 dense_19 (Dense)            (None, 3)                 387

=================================================================
Total params: 1,626,211
Trainable params: 1,626,211
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
import tensorflow as tf
from tensorflow.keras import layers, models

train_dir = '/Train_Folder_out/'
validation_dir = '/Test_Folder_out/'

img_size = (64, 64)
batch_size = 32

train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=img_size,
    batch_size=batch_size,
)
validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    validation_dir,
    image_size=img_size,
    batch_size=batch_size,
)
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.AveragePooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.AveragePooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(3, activation='softmax')
])
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
    "best_model.h5",
    monitor='val_loss',
    save_best_only=True
)
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=[early_stopping, model_checkpoint]
)
best_model = tf.keras.models.load_model("best_model.h5")
test_loss, test_acc = best_model.evaluate(validation_dataset)
print('Test accuracy:', test_acc)
best_model.summary()
```

```
Found 1200 files belonging to 3 classes.
Found 487 files belonging to 3 classes.
Epoch 1/10
38/38 [==============================] - 6s 122ms/step - loss: 3.8176 - accuracy:
0.8808 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 2/10
38/38 [==============================] - 5s 121ms/step - loss: 7.3711e-05 - accura
cy: 1.0000 - val_loss: 9.4498e-05 - val_accuracy: 1.0000
Epoch 3/10
38/38 [==============================] - 5s 120ms/step - loss: 1.4453e-05 - accura
cy: 1.0000 - val_loss: 8.3171e-06 - val_accuracy: 1.0000
Epoch 4/10
38/38 [==============================] - 5s 118ms/step - loss: 6.9727e-07 - accura
cy: 1.0000 - val_loss: 1.1828e-05 - val_accuracy: 1.0000
Epoch 5/10
38/38 [==============================] - 5s 118ms/step - loss: 4.9988e-07 - accura
cy: 1.0000 - val_loss: 1.4547e-05 - val_accuracy: 1.0000
Epoch 6/10
38/38 [==============================] - 5s 120ms/step - loss: 4.1465e-07 - accura
cy: 1.0000 - val_loss: 1.7004e-05 - val_accuracy: 1.0000
16/16 [==============================] - 1s 30ms/step - loss: 8.3171e-06 - accurac
y: 1.0000
Test accuracy: 1.0
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 62, 62, 32)        896

 average_pooling2d_13 (Avera  (None, 31, 31, 32)       0
 gePooling2D)

 conv2d_21 (Conv2D)          (None, 29, 29, 64)        18496

 average_pooling2d_14 (Avera  (None, 14, 14, 64)       0
 gePooling2D)

 conv2d_22 (Conv2D)          (None, 12, 12, 128)       73856

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 flatten_7 (Flatten)         (None, 4608)              0

 dense_20 (Dense)            (None, 128)               589952

 dense_21 (Dense)            (None, 64)                8256

 dense_22 (Dense)            (None, 3)                 195

=================================================================
Total params: 691,651
Trainable params: 691,651
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import classification_report

best_model = tf.keras.models.load_model("best_model.h5")
validation_loss, validation_accuracy = best_model.evaluate(validation_dataset, verb
print('Validation accuracy:', validation_accuracy)

# Save the best model to a custom location
save_path = "/examples/"
best_model.save(save_path)

true_labels = []
predicted_labels = []
for images, labels in validation_dataset:
    true_labels.extend(labels.numpy())
    predicted_labels.extend(tf.argmax(best_model.predict(images), axis=-1).numpy())
report = classification_report(true_labels, predicted_labels)

print("Classification Report:")
print(report)
```

```
16/16 - 1s - loss: 8.3171e-06 - accuracy: 1.0000 - 917ms/epoch - 57ms/step
Validation accuracy: 1.0
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_c
ompiled_convolution_op, _jit_compiled_convolution_op, _update_step_xla while savin
g (showing 4 of 4). These functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: /examples/assets
```

```
INFO:tensorflow:Assets written to: /examples/assets
```

```
1/1 [==============================] - 0s 86ms/step
1/1 [==============================] - 0s 86ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 72ms/step
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       164
           1       1.00      1.00      1.00       166
           2       1.00      1.00      1.00       157

    accuracy                           1.00       487
   macro avg       1.00      1.00      1.00       487
weighted avg       1.00      1.00      1.00       487
```

In [ ]:

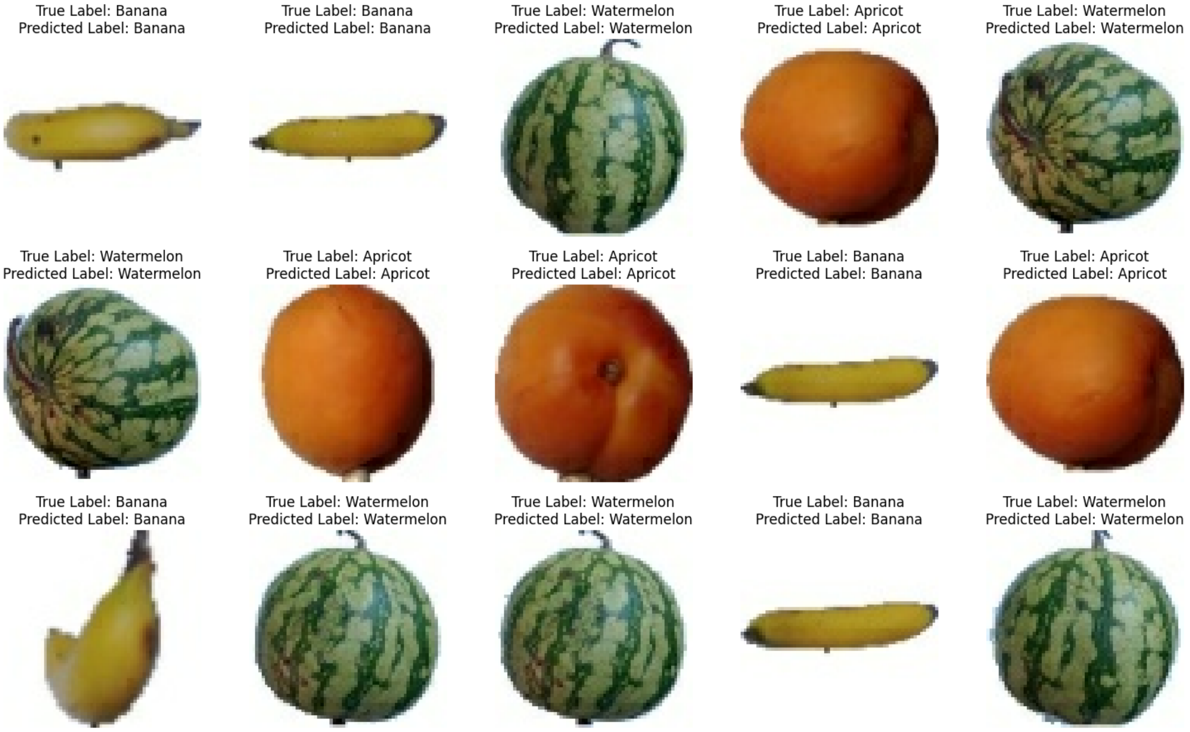
```
import matplotlib.pyplot as plt
import numpy as np
num_samples = 15
test_images = []
test_labels = []
for images, labels in validation_dataset.take(num_samples):
    test_images.append(images[0])
    test_labels.append(labels[0].numpy())

if len(test_images) < num_samples:
    num_samples = len(test_images)
    print(f"Reduced num_samples to {num_samples} because the dataset contains fewer

predicted_labels = best_model.predict(np.array(test_images))
predicted_labels = tf.argmax(predicted_labels, axis=-1).numpy()

class_names = train_dataset.class_names
if num_samples > 0:
    plt.figure(figsize=(15, 15))
    for i in range(num_samples):
        plt.subplot(5, 5, i+1)
        plt.imshow(test_images[i].numpy().astype("uint8"))
        true_label = class_names[test_labels[i]]
        predicted_label = class_names[predicted_labels[i]]
        plt.title(f"True Label: {true_label}\nPredicted Label: {predicted_label}")
        plt.axis('off')
    plt.tight_layout()
    plt.show()
else:
    print("No samples to plot.")
```

```
1/1 [==============================] - 0s 35ms/step
```


True Label: Banana
Predicted Label: Banana

True Label: Banana
Predicted Label: Banana

True Label: Watermelon
Predicted Label: Watermelon

True Label: Apricot
Predicted Label: Apricot

True Label: Watermelon
Predicted Label: Watermelon

True Label: Watermelon
Predicted Label: Watermelon

True Label: Apricot
Predicted Label: Apricot

True Label: Apricot
Predicted Label: Apricot

True Label: Banana
Predicted Label: Banana

True Label: Apricot
Predicted Label: Apricot

True Label: Banana
Predicted Label: Banana

True Label: Watermelon
Predicted Label: Watermelon

True Label: Watermelon
Predicted Label: Watermelon

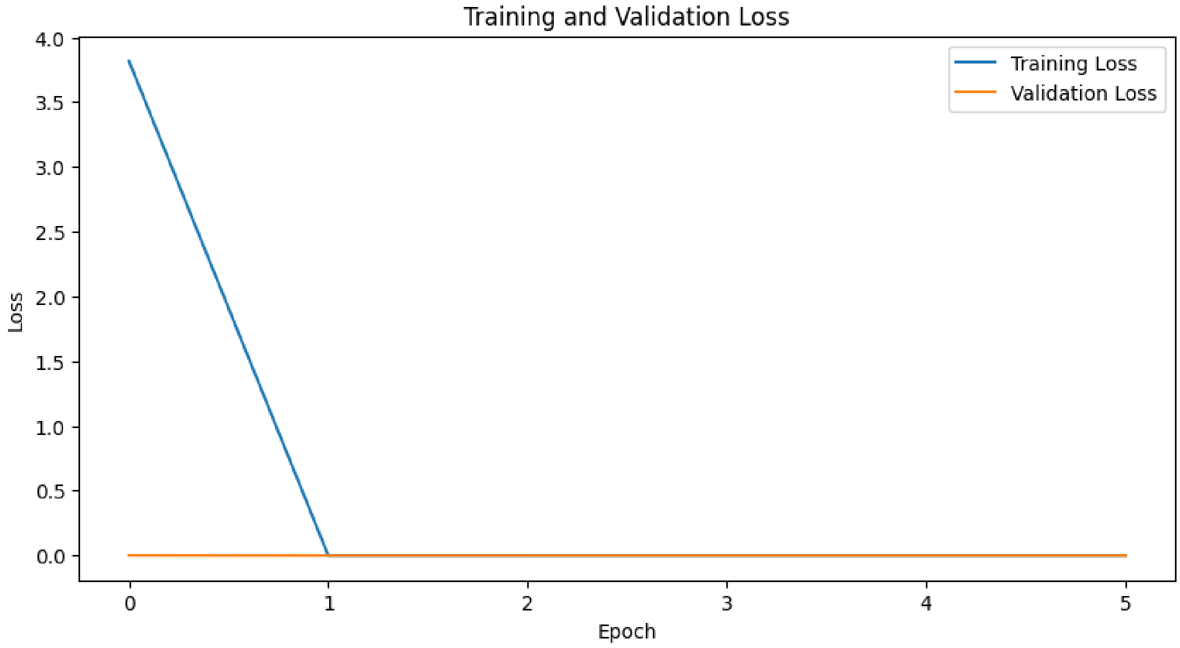
True Label: Banana
Predicted Label: Banana

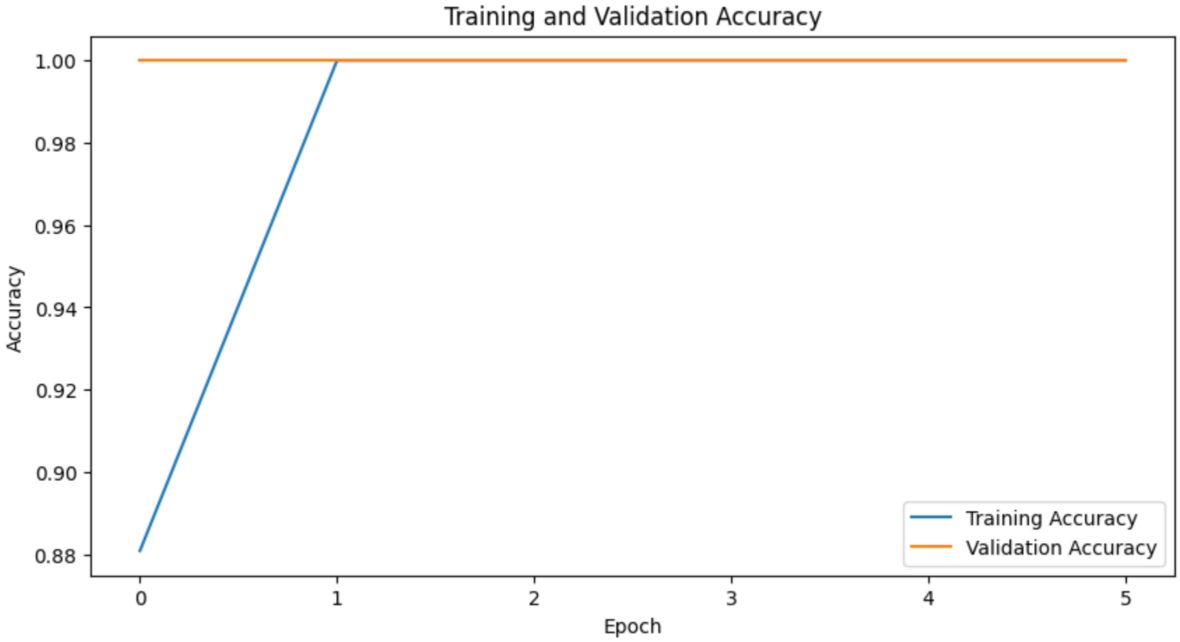
True Label: Watermelon
Predicted Label: Watermelon

In [ ]:

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
best_model = tf.keras.models.load_model("best_model.h5")
validation_loss, validation_accuracy = best_model.evaluate(validation_dataset)
print('Validation accuracy:', validation_accuracy)
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
16/16 [==============================] - 1s 32ms/step - loss: 8.3171e-06 - accurac
y: 1.0000
Validation accuracy: 1.0
```


Training and Validation Loss
Training Loss
Validation Loss
Loss
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
Epoch


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
1
2
3
4
5
Epoch