```
import tensorflow as tf
from tensorflow import keras

img_height =224
img_width =224
batch_size = 32
validation_split = 0.2


rescale = tf.keras.layers.Rescaling(1./255)


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/drive/MyDrive/AIandML/worksheet6/FruitinAmazon.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()


# data.data.districts
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout


train_dir = '/content/FruitinAmazon/train'


train_ds = tf.keras.preprocessing.image_dataset_from_directory(
train_dir,
labels='inferred',
label_mode='int',
image_size=(img_height, img_width),
interpolation='nearest',
batch_size=batch_size,
shuffle=True,
validation_split=validation_split,
subset='training',
seed=123
)
train_ds = train_ds.map(lambda x, y: (rescale(x), y))
```

```
Found 90 files belonging to 6 classes.
Using 72 files for training.
```

```
# Create validation dataset with normalization
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
train_dir,
labels='inferred',
label_mode='int',
image_size=(img_height, img_width),
interpolation='nearest',
batch_size=batch_size,
shuffle=False,
validation_split=validation_split,
subset='validation',
seed=123
)
val_ds = val_ds.map(lambda x, y: (rescale(x), y))
```

```
Found 90 files belonging to 6 classes.
Using 18 files for validation.
```

```
from tensorflow.keras.applications import VGG16
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))


base_model.summary()
```

Model: "vgg16"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_2 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1,792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36,928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73,856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147,584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295,168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1,180,160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |

Total params: 14,714,688 (56.13 MB)
Trainable params: 14,714,688 (56.13 MB)

```
for layer in base_model.layers:
  layer.trainable = False
```

```
model = Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(6, activation='softmax'))
```

```
model.summary()
```

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| vgg16 (Functional) | (None, 7, 7, 512) | 14,714,688 |
| flatten_1 (Flatten) | (None, 25088) | 0 |
| dense_2 (Dense) | (None, 1024) | 25,691,136 |
| dense_3 (Dense) | (None, 6) | 6,150 |

Total params: 40,411,974 (154.16 MB)
Trainable params: 25,697,286 (98.03 MB)

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
history = model.fit(train_ds, validation_data=val_ds, epochs=10)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 67s 23s/step - accuracy: 0.6046 - loss: 2.4378 - val_accuracy: 0.2778 - val_loss: 6.7570
Epoch 4/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 58s 18s/step - accuracy: 0.6771 - loss: 1.4654 - val_accuracy: 0.6111 - val_loss: 2.5162
Epoch 5/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 82s 19s/step - accuracy: 0.9340 - loss: 0.3740 - val_accuracy: 0.8889 - val_loss: 0.4978
Epoch 6/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 90s 23s/step - accuracy: 0.9410 - loss: 0.1172 - val_accuracy: 0.7778 - val_loss: 0.9794
Epoch 7/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 67s 23s/step - accuracy: 0.8485 - loss: 0.3758 - val_accuracy: 0.8889 - val_loss: 0.5040
Epoch 8/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 67s 23s/step - accuracy: 1.0000 - loss: 0.0071 - val_accuracy: 0.9444 - val_loss: 0.3117
Epoch 9/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 57s 18s/step - accuracy: 1.0000 - loss: 0.0015 - val_accuracy: 0.9444 - val_loss: 0.4493
Epoch 10/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 58s 18s/step - accuracy: 1.0000 - loss: 0.0109 - val_accuracy: 0.9444 - val_loss: 0.5505
Epoch 1/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 81s 27s/step - accuracy: 0.1437 - loss: 13.1538 - val_accuracy: 0.8333 - val_loss: 2.8701
Epoch 2/10
```

```
test_loss, test_acc = model.evaluate(val_ds)
print(f"Validation Accuracy: {test_acc:.2f}")
```

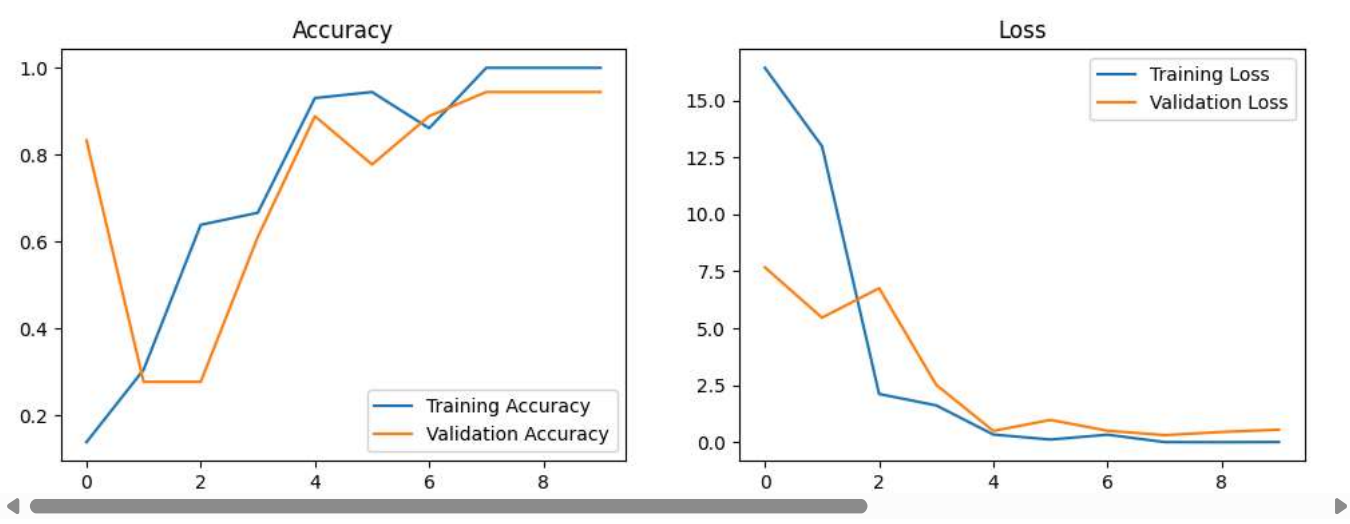
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step - accuracy: 0.9444 - loss: 0.5505
Validation Accuracy: 0.94
```

```
#plot for test data
# Plot training history
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()

plt.show()
```




```
test_dir = "/content/FruitinAmazon/test"
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size,
    shuffle=False,
    interpolation='nearest',
    seed=123
)
```

```
test_ds = test_ds.map(lambda x, y: (rescale(x), y))
```

```
---------------------------------------------------------------------------
NotFoundError                             Traceback (most recent call last)
<ipython-input-29-777ca6433590> in <cell line: 0>()
      1 test_dir = "/content/drive/MyDrive/Level 6/Artificial_Intelligence/Week5/FruitinAmazon/test"
----> 2 test_ds = tf.keras.preprocessing.image_dataset_from_directory(
      3     test_dir,
      4     labels='inferred',
      5     label_mode='int',

2 frames
/usr/local/lib/python3.11/dist-packages/tensorflow/python/lib/io/file_io.py in list_directory_v2(path)
    766   """
    767   if not is_directory(path):
--> 768     raise errors.NotFoundError(
    769         node_def=None,
    770         op=None,

NotFoundError: Could not find directory /content/drive/MyDrive/Level 6/Artificial_Intelligence/Week5/FruitinAmazon/test
```

Next steps: Explain error

```
test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Accuracy: {test_accuracy:.4f}")
print(f"Test Loss: {test_loss:.4f}")


import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Flatten
from keras.applications.vgg16 import VGG16
from tensorflow.keras.optimizers import RMSprop


base_model = VGG16(
    weights='imagenet',
    include_top = False,
    input_shape=(224,224,3)
)


base_model.trainable = True

set_trainable = False

for layer in base_model.layers:
  if layer.name == 'block5_conv1':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

for layer in base_model.layers:
  print(layer.name,layer.trainable)


base_model.summary()


model = Sequential()

model.add(base_model)
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dense(6,activation='softmax'))


model.compile(optimizer=keras.optimizers.RMSprop(learning_rate=1e-5), loss='sparse_categorical_crossentropy', metrics=['accuracy'])


test_loss, test_acc = model.evaluate(val_ds)
print(f"Validation Accuracy: {test_acc:.2f}")


#plot for test data
# Plot training history
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
```

```
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()

plt.show()



test_dir = "/content/drive/MyDrive/Level 6/Artificial_Intelligence/Week5/FruitinAmazon/test"

test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size,
    shuffle=False,
    interpolation='nearest',
    seed=123
)


test_ds = test_ds.map(lambda x, y: (rescale(x), y))


test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Accuracy: {test_accuracy:.4f}")
print(f"Test Loss: {test_loss:.4f}")


Start coding or generate with AI.
```