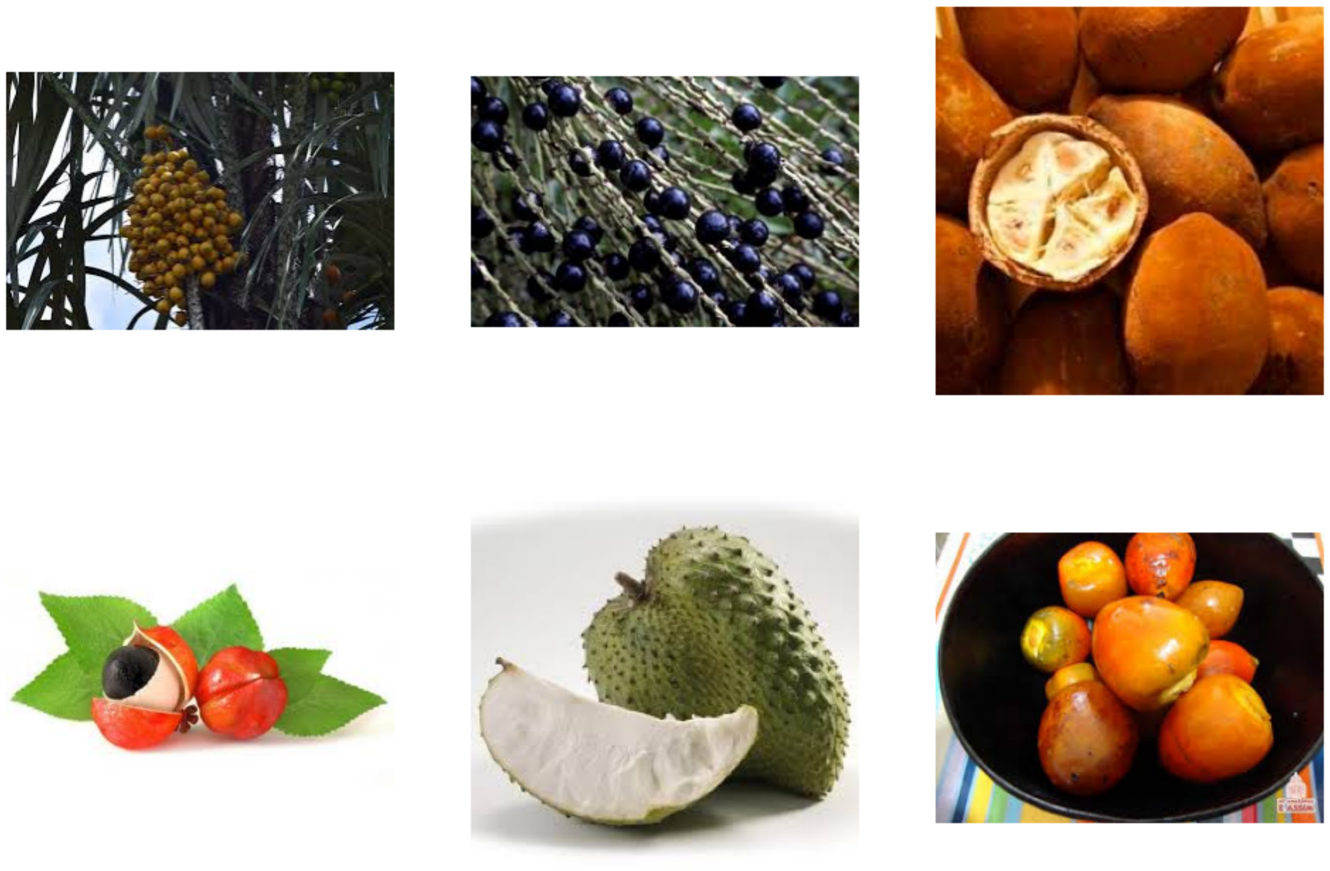
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import os
import random
from PIL import Image
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```

```python
train_dir = '/content/drive/MyDrive/AI and ML/week_five/FruitinAmazon/train'
test_dir = '/content/drive/MyDrive/AI and ML/week_five/FruitinAmazon/test'
```

```
 class_dirs = os.listdir(train_dir)

images = []
for class_dir in class_dirs:
    class_path = os.path.join(train_dir, class_dir)
    image_path = random.choice(os.listdir(class_path))
    img = Image.open(os.path.join(class_path, image_path))
    images.append(img)

fig, axes = plt.subplots(2, len(images)//2, figsize=(12, 8))
for ax, img in zip(axes.flatten(), images):
    ax.imshow(img)
    ax.axis('off')
plt.show()
```



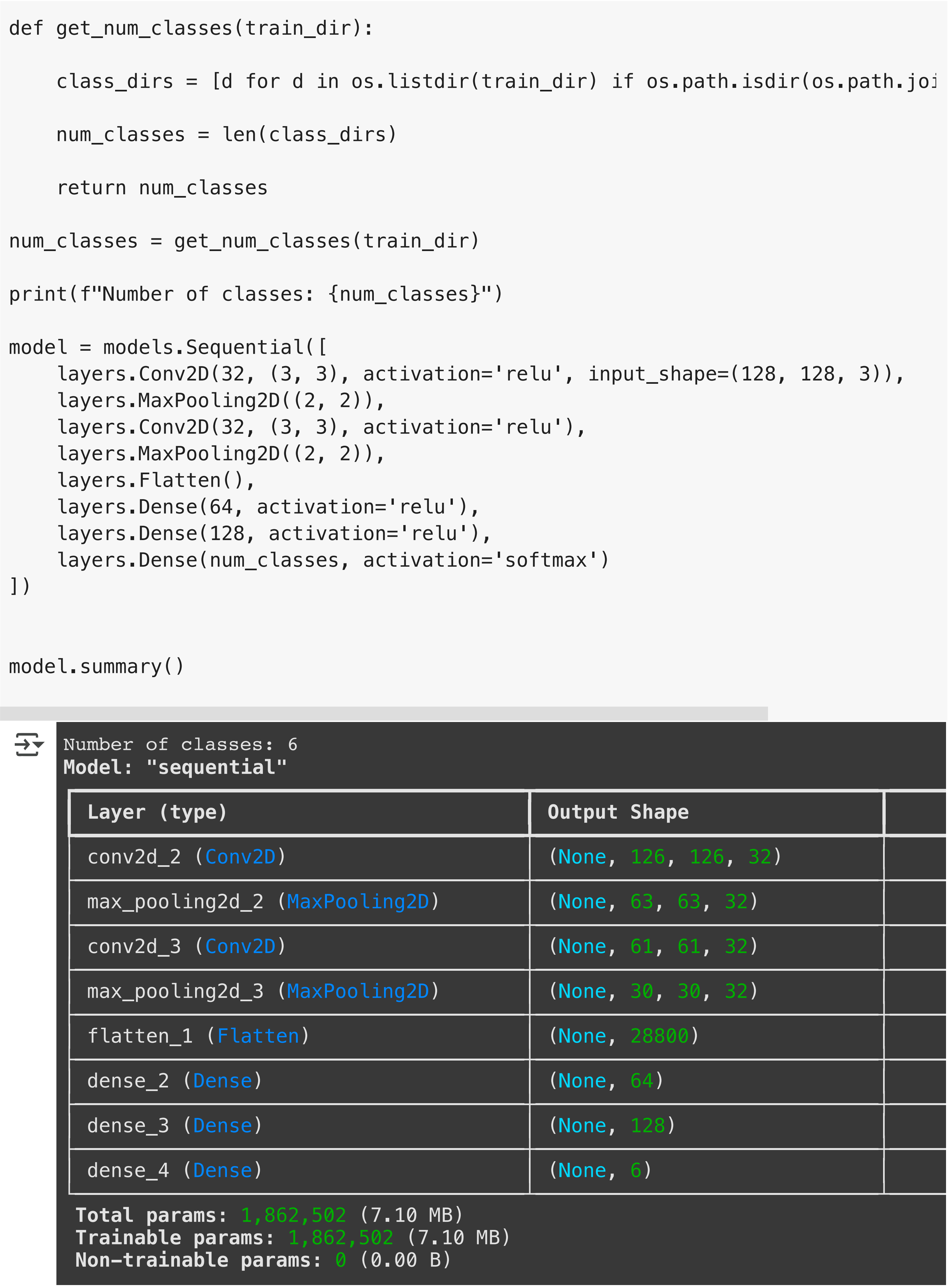
```
class_dirs = os.listdir(test_dir)

images = []
for class_dir in class_dirs:
    class_path = os.path.join(train_dir, class_dir)
    image_path = random.choice(os.listdir(class_path))
    img = Image.open(os.path.join(class_path, image_path))
    images.append(img)

fig, axes = plt.subplots(2, len(images)//2, figsize=(12, 8))
for ax, img in zip(axes.flatten(), images):
    ax.imshow(img)
    ax.axis('off')
plt.show()
```

```
def check_for_corrupted_images(train_dir):
    removed_images = []
    for class_dir in os.listdir(train_dir):
        class_path = os.path.join(train_dir, class_dir)
        for image_name in os.listdir(class_path):
            image_path = os.path.join(class_path, image_name)
            try:
                img = Image.open(image_path)
                img.verify()
            except (IOError, SyntaxError) as e:
                removed_images.append(image_path)
                os.remove(image_path)
                print(f"Removed corrupted image: {image_path}")

    if not removed_images:
        print("No corrupted images found.")

check_for_corrupted_images(train_dir)
```

```
No corrupted images found.
```

```
img_height = 128
img_width = 128
batch_size = 32
validation_split = 0.2

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

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size,
    validation_split=validation_split,
    subset='training',
    seed=123
)

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

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size,
    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 72 files for training.
Found 90 files belonging to 6 classes.
Using 18 files for validation.
```



```
def get_num_classes(train_dir):

    class_dirs = [d for d in os.listdir(train_dir) if os.path.isdir(os.path.joi

    num_classes = len(class_dirs)

    return num_classes

num_classes = get_num_classes(train_dir)

print(f"Number of classes: {num_classes}")

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])


model.summary()
```

```
Number of classes: 6
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━
┃ Layer (type)                         ┃ Output Shape                ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━
│ conv2d_2 (Conv2D)                    │ (None, 126, 126, 32)        │
├──────────────────────────────────────┼─────────────────────────────┼─
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 63, 63, 32)          │
├──────────────────────────────────────┼─────────────────────────────┼─
│ conv2d_3 (Conv2D)                    │ (None, 61, 61, 32)          │
├──────────────────────────────────────┼─────────────────────────────┼─
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 30, 30, 32)          │
├──────────────────────────────────────┼─────────────────────────────┼─
│ flatten_1 (Flatten)                  │ (None, 28800)               │
├──────────────────────────────────────┼─────────────────────────────┼─
│ dense_2 (Dense)                      │ (None, 64)                  │
├──────────────────────────────────────┼─────────────────────────────┼─
│ dense_3 (Dense)                      │ (None, 128)                 │
├──────────────────────────────────────┼─────────────────────────────┼─
│ dense_4 (Dense)                      │ (None, 6)                   │
└──────────────────────────────────────┴─────────────────────────────┴─
 Total params: 1,862,502 (7.10 MB)
 Trainable params: 1,862,502 (7.10 MB)
 Non-trainable params: 0 (0.00 B)
```

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # For multi-class as we
              metrics=['accuracy'])
```

```
callbacks = [
    ModelCheckpoint('best_model.h5', save_best_only=True, monitor='val_loss'),
    EarlyStopping(monitor='val_loss', patience=4)
]

history = model.fit(
    train_ds,
    epochs=250,
    validation_data=val_ds,
    callbacks=callbacks,
    batch_size=16
)
```

```
Epoch 1/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 425ms/step - accuracy: 0.6910 - loss: 0.8075WAR
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 620ms/step - accuracy: 0.6849 - loss: 0.8148 -
Epoch 2/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 355ms/step - accuracy: 0.6881 - loss: 0.7535WAR
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 578ms/step - accuracy: 0.7001 - loss: 0.7400 -
Epoch 3/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 350ms/step - accuracy: 0.9404 - loss: 0.4746WAR
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 487ms/step - accuracy: 0.9379 - loss: 0.4759 -
Epoch 4/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 781ms/step - accuracy: 0.9566 - loss: 0.2935 -
Epoch 5/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 525ms/step - accuracy: 0.9714 - loss: 0.2220 -
Epoch 6/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 447ms/step - accuracy: 1.0000 - loss: 0.1382 -
Epoch 7/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 512ms/step - accuracy: 0.9674 - loss: 0.1527 -
```

```
img_height = 128
img_width = 128
batch_size = 32

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
)


rescale = tf.keras.layers.Rescaling(1./255)
test_ds = test_ds.map(lambda x, y: (rescale(x), y))


test_loss, test_acc = model.evaluate(test_ds)
print(f"Test accuracy: {test_acc:.4f}")
```

```
Found 30 files belonging to 6 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 429ms/step - accuracy: 0.6333 - loss: 0.9617
Test accuracy: 0.6333
```

```
model.save('cnn_model.h5')

loaded_model = tf.keras.models.load_model('cnn_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` o
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet t
```