Worksheet 5 output

Task 1: Data Understanding and Visualization:

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

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

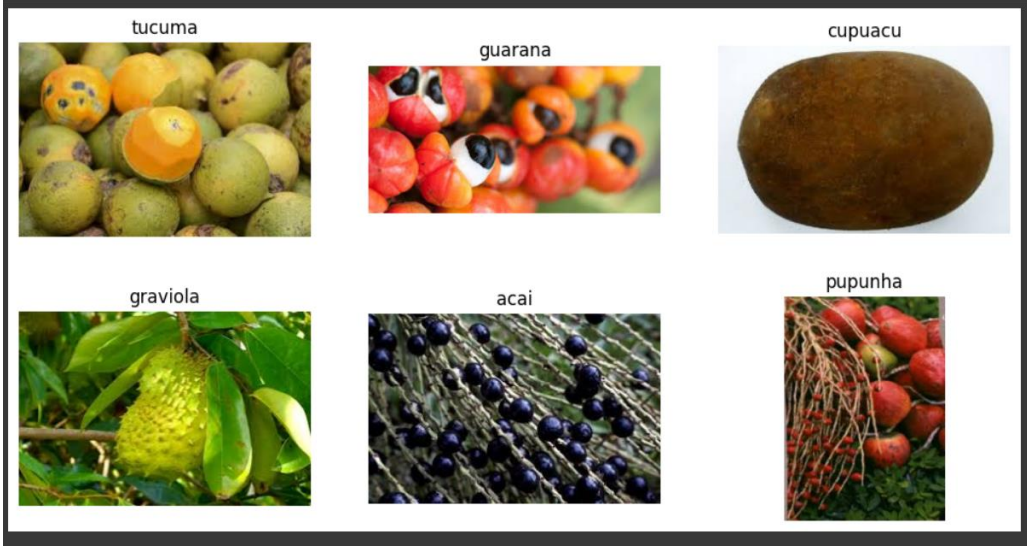
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from PIL import Image
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
import random
```

```
train_path = "/content/drive/MyDrive/Artificial Intelligence and machine learning/FruitinAmazon/FruitinAmazon/train"
test_path = "/content/drive/MyDrive/Artificial Intelligence and machine learning/FruitinAmazon/FruitinAmazon/test"
```

```
os.listdir(train_path)
```

```
['tucuma', 'guarana', 'cupuacu', 'graviola', 'acai', 'pupunha']
```

```
visualise(train_path)
```



```
            img.verify()
        except Exception as e:
            print(f"Corrupted image found {image}")
    print("No corrupted image found")

check_for_corrupted(train_path)
```

```
No corrupted image found
```

+ Code + Text

Task 2: Loading and Preprocessing Image Data in keras:

+ Code + Text

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

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

train_ds = tf.keras.utils.image_dataset_from_directory(
    train_path,
    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
```

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

```
Found 90 files belonging to 6 classes.
Using 72 files for training.
Found 30 files belonging to 6 classes.
Found 30 files belonging to 6 classes.
```

+ Code + Text

Task 3 - Implement a CNN with Convolutional Architecture:

```
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), padding='same', strides=1, activation='relu', input_shape=(128, 128, 3)),
    layers.MaxPooling2D((2, 2), strides=2),
    layers.Conv2D(32, (3, 3), padding='same', strides=1, activation='relu'),
    layers.MaxPooling2D((2, 2), strides=2),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(6, activation="softmax")
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seque
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 32)	0
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 64)	2,097,216
dense_1 (Dense)	(None, 128)	8,320
dense_2 (Dense)	(None, 6)	774

Total params: 2,116,454 (8.07 MB)
Trainable params: 2,116,454 (8.07 MB)
Non-trainable params: 0 (0.00 B)

Task 4: Compile the Model

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

```
    batch_size=16,
    validation_data=val_ds,
    callbacks=callbacks
)
```

```
Epoch 1/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 15s 6s/step - accuracy: 0.1623 - loss: 1.8829 - val_accuracy: 0.3333 - val_loss: 1.7234
Epoch 2/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 821ms/step - accuracy: 0.4062 - loss: 1.6800 - val_accuracy: 0.3333 - val_loss: 1.6413
Epoch 3/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 604ms/step - accuracy: 0.3958 - loss: 1.5225 - val_accuracy: 0.1667 - val_loss: 1.7050
Epoch 4/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 996ms/step - accuracy: 0.3832 - loss: 1.3420 - val_accuracy: 0.5333 - val_loss: 1.3762
Epoch 5/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 0.6337 - loss: 1.0028 - val_accuracy: 0.3333 - val_loss: 1.3530
Epoch 6/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 649ms/step - accuracy: 0.6771 - loss: 0.8534 - val_accuracy: 0.4000 - val_loss: 1.4508
Epoch 7/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 619ms/step - accuracy: 0.7422 - loss: 0.7113 - val_accuracy: 0.4000 - val_loss: 1.3276
Epoch 8/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 701ms/step - accuracy: 0.8906 - loss: 0.5053 - val_accuracy: 0.6333 - val_loss: 1.0230
Epoch 9/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 952ms/step - accuracy: 0.9488 - loss: 0.2670 - val_accuracy: 0.7000 - val_loss: 0.9795
Epoch 10/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 667ms/step - accuracy: 0.9705 - loss: 0.1708 - val_accuracy: 0.6333 - val_loss: 1.3013
Epoch 11/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 630ms/step - accuracy: 0.9410 - loss: 0.1909 - val_accuracy: 0.6667 - val_loss: 1.0288
Epoch 12/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 666ms/step - accuracy: 0.9518 - loss: 0.1235 - val_accuracy: 0.5667 - val_loss: 1.2649
Epoch 13/250
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 702ms/step - accuracy: 0.9891 - loss: 0.0734 - val_accuracy: 0.5333 - val_loss: 1.2460
```

```
plt.show()
```

Task 5: Evaluate the Model

+ Code + Text

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 329ms/step - accuracy: 0.7000 - loss: 0.9795
Test Accuracy: 0.7000
```

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

bestest_model = tf.keras.models.load_model('bestest_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

Task 7: Predictions and Classification Report

```
import numpy as np
from sklearn.metrics import classification_report

y_pred_prob = bestest_model.predict(test_ds)
y_pred = np.argmax(y_pred_prob, axis=1)

y_true = []
for images, labels in test_ds.unbatch():
```

```
y_true = np.array(y_true)
print(y_true)
print(classification_report(y_true, y_pred))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 689ms/step
[0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5]
              precision    recall  f1-score   support

           0       0.67      0.80      0.73         5
           1       0.50      0.60      0.55         5
           2       0.71      1.00      0.83         5
           3       0.67      0.80      0.73         5
           4       1.00      0.60      0.75         5
           5       1.00      0.40      0.57         5

    accuracy                           0.70        30
   macro avg       0.76      0.70      0.69        30
weighted avg       0.76      0.70      0.69        30
```