Worksheet-6

```
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_remoun
```

Importing Necessary Tools

```
import os
import random
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import load_img, ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.regularizers import l2
from sklearn.metrics import classification_report
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Input
from tensorflow.keras.models import Model
```

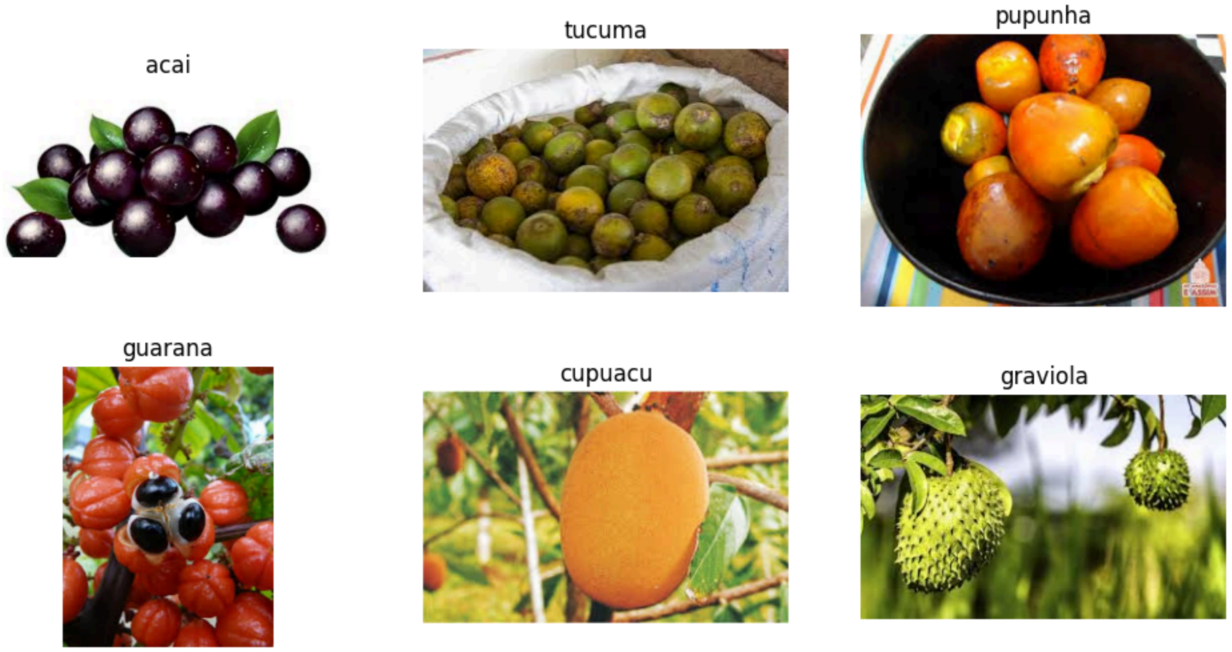
Task-1

```
train_dir = "/content/drive/MyDrive/AI and ML/Week5/FruitinAmazon/train"
test_dir = "/content/drive/MyDrive/AI and ML/Week5/FruitinAmazon/test"
```

```
class_names = os.listdir(train_dir)
print(f"Classes: {class_names}")
```

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

```
def visualize_images(train_dir, class_names):
    fig, axes = plt.subplots(2, len(class_names) // 2, figsize=(12, 6))
    axes = axes.flatten()
    for i, class_name in enumerate(class_names):
        class_path = os.path.join(train_dir, class_name)
        img_name = random.choice(os.listdir(class_path))
        img_path = os.path.join(class_path, img_name)
        img = load_img(img_path)
        axes[i].imshow(img)
        axes[i].set_title(class_name)
        axes[i].axis("off")
    plt.show()

visualize_images(train_dir, class_names)
```



```
damagedImages = []
for class_name in class_names:
    class_path = os.path.join(train_dir, class_name)
    for img_name in os.listdir(class_path):
        img_path = os.path.join(class_path, img_name)
        try:
            img = load_img(img_path)  # Try opening the image
        except (IOError, SyntaxError):
            damagedImages.append(img_path)
            os.remove(img_path)
            print(f"Damaged image removed: {img_path}")

if not damagedImages:
    print("No Damaged Images Found.")
```

```
No Damaged Images Found.
```

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


train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=validation_split,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2]
)

val_datagen = ImageDataGenerator(rescale=1./255, validation_split=validation_split)


train_ds = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='sparse',
    subset='training',
    shuffle=True,
    seed=123
)
```

```
Found 74 images belonging to 6 classes.
```

```
val_ds = val_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='sparse',
    subset='validation',
    shuffle=False,
    seed=123
)
```

```
Found 18 images belonging to 6 classes.
```

```
num_classes = len(class_names)

model = Sequential([
    Conv2D(32, (3,3), activation='relu', padding='same', kernel_regularizer=l2(0.001), input_shape=(img_height, img_width, 3
    BatchNormalization(),
    Conv2D(64, (3,3), activation='relu', padding='same', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    MaxPooling2D((2,2)),
    Dropout(0.25),

    Conv2D(128, (3,3), activation='relu', padding='same', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    MaxPooling2D((2,2)),
    Dropout(0.4),

    Flatten(),
    Dense(256, activation='relu', kernel_regularizer=l2(0.001)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `in
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 128, 128, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 128, 128, 32)   │           128 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 128, 128, 64)   │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 128, 128, 64)   │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 64, 64, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 64, 64, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 64, 64, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 64, 64, 128)    │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 32, 32, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 32, 32, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 131072)         │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 256)            │    33,554,688 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 256)            │         1,024 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_2 (Dropout)             │ (None, 256)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 6)              │         1,542 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 33,651,398 (128.37 MB)
 Trainable params: 33,650,438 (128.37 MB)
 Non-trainable params: 960 (3.75 KB)
```

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

```
callbacks = [
    ModelCheckpoint("best_model.h5", save_best_only=True, monitor="val_accuracy", mode="max"),
    EarlyStopping(monitor="val_loss", patience=5, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
]
```

```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=30,
    batch_size=16,
    callbacks=callbacks
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `P
  self._warn_if_super_not_called()
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.2248 - loss: 3.5524WARNING:absl:You are saving your model as an HDF5 f
3/3 ━━━━━━━━━━━━━━━━━━━━ 23s 5s/step - accuracy: 0.2395 - loss: 3.5551 - val_accuracy: 0.1667 - val_loss: 2.8044 - learn
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 18s 5s/step - accuracy: 0.4584 - loss: 2.4822 - val_accuracy: 0.1667 - val_loss: 2.7587 - learn
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 5s/step - accuracy: 0.5482 - loss: 2.2836WARNING:absl:You are saving your model as an HDF5 f
3/3 ━━━━━━━━━━━━━━━━━━━━ 17s 6s/step - accuracy: 0.5429 - loss: 2.3286 - val_accuracy: 0.2222 - val_loss: 3.4875 - learn
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 17s 5s/step - accuracy: 0.5192 - loss: 2.7527 - val_accuracy: 0.1667 - val_loss: 5.6354 - learn
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 16s 4s/step - accuracy: 0.5874 - loss: 2.2804 - val_accuracy: 0.1667 - val_loss: 9.6408 - learn
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 17s 5s/step - accuracy: 0.5554 - loss: 2.5370 - val_accuracy: 0.1667 - val_loss: 14.3421 - lear
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 17s 7s/step - accuracy: 0.6609 - loss: 2.5455 - val_accuracy: 0.1667 - val_loss: 19.0799 - lear
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_ds = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='sparse',
    shuffle=False
)
test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
Found 38 images belonging to 6 classes.
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 308ms/step - accuracy: 0.1678 - loss: 2.7045
Test Accuracy: 15.79%
```

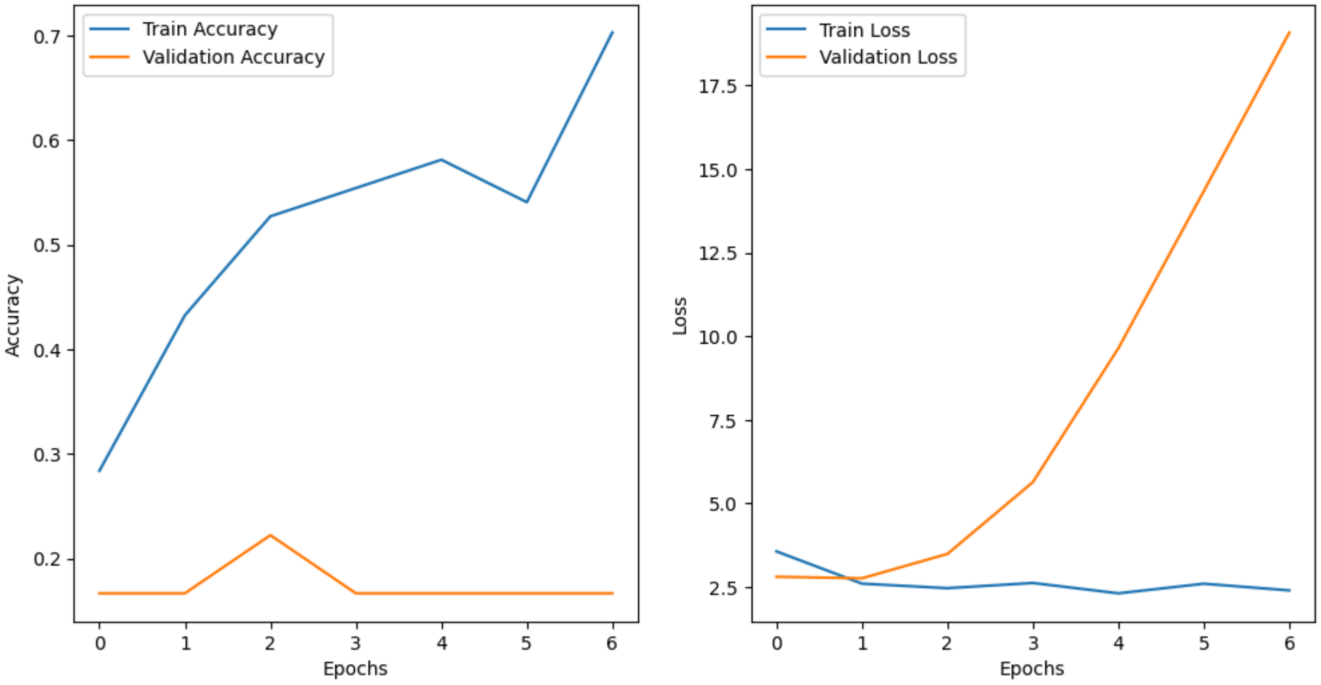
```
model.save("final_model.h5")
loaded_model = tf.keras.models.load_model("final_model.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
```

```
y_true = test_ds.classes
y_pred = np.argmax(loaded_model.predict(test_ds), axis=1)

print(classification_report(y_true, y_pred, target_names=class_names))
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 330ms/step
              precision    recall  f1-score   support

        acai       0.23      0.60      0.33         5
      tucuma       0.00      0.00      0.00         5
     pupunha       0.00      0.00      0.00         5
     guarana       0.00      0.00      0.00         5
     cupuacu       0.33      0.38      0.35         8
    graviola       0.00      0.00      0.00        10

    accuracy                           0.16        38
   macro avg       0.09      0.16      0.11        38
weighted avg       0.10      0.16      0.12        38

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Task-2

```
base_model = MobileNetV2(input_shape=(img_height, img_width, 3), include_top=False, weights='imagenet')
base_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
inputs = Input(shape=(img_height, img_width, 3))
x = base_model(inputs, training=False)
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.4)(x)
outputs = Dense(num_classes, activation='softmax')(x)
model = Model(inputs, outputs)

model.summary()
```

```
Model: "functional_16"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer_2 (InputLayer)      │ (None, 128, 128, 3)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ mobilenetv2_1.00_128            │ (None, 4, 4, 1280)     │     2,257,984 │
│ (Functional)                    │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling2d        │ (None, 1280)           │             0 │
│ (GlobalAveragePooling2D)        │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 128)            │       163,968 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_3 (Dropout)             │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 6)              │           774 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 2,422,726 (9.24 MB)
 Trainable params: 164,742 (643.52 KB)
 Non-trainable params: 2,257,984 (8.61 MB)
```

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

```
callbacks = [
    ModelCheckpoint("best_model_tl.h5", save_best_only=True, monitor="val_accuracy", mode="max"),
    EarlyStopping(monitor="val_loss", patience=5, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
]

# Train the model (only top layers)
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=30,
    callbacks=callbacks
)
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 579ms/step - accuracy: 0.0588 - loss: 3.1303WARNING:absl:You are saving your model as an HDF
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 2s/step - accuracy: 0.0644 - loss: 3.0073 - val_accuracy: 0.3333 - val_loss: 1.6351 - learni
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 432ms/step - accuracy: 0.3663 - loss: 1.4966WARNING:absl:You are saving your model as an HDF
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 969ms/step - accuracy: 0.3761 - loss: 1.4908 - val_accuracy: 0.5556 - val_loss: 1.1324 - lea
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.6209 - loss: 1.1606WARNING:absl:You are saving your model as an HDF5 f
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 1s/step - accuracy: 0.6076 - loss: 1.1603 - val_accuracy: 0.7222 - val_loss: 0.8835 - learni
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 737ms/step - accuracy: 0.6736 - loss: 0.8691 - val_accuracy: 0.7222 - val_loss: 0.7755 - lea
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 714ms/step - accuracy: 0.8379 - loss: 0.5938 - val_accuracy: 0.7222 - val_loss: 0.7042 - lea
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 552ms/step - accuracy: 0.8647 - loss: 0.5295 - val_accuracy: 0.7222 - val_loss: 0.6147 - lea
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 288ms/step - accuracy: 0.7620 - loss: 0.5701WARNING:absl:You are saving your model as an HDF
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 563ms/step - accuracy: 0.7640 - loss: 0.5637 - val_accuracy: 0.8333 - val_loss: 0.5537 - lea
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 934ms/step - accuracy: 0.8217 - loss: 0.4723 - val_accuracy: 0.8333 - val_loss: 0.5021 - lea
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 822ms/step - accuracy: 0.9454 - loss: 0.2832 - val_accuracy: 0.7778 - val_loss: 0.4686 - lea
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 522ms/step - accuracy: 0.8976 - loss: 0.3698 - val_accuracy: 0.8333 - val_loss: 0.4511 - lea
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 430ms/step - accuracy: 0.9165 - loss: 0.3067 - val_accuracy: 0.8333 - val_loss: 0.4383 - lea
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 528ms/step - accuracy: 0.9727 - loss: 0.1493 - val_accuracy: 0.8333 - val_loss: 0.4284 - lea
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 707ms/step - accuracy: 0.9428 - loss: 0.2147 - val_accuracy: 0.7778 - val_loss: 0.4398 - lea
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 837ms/step - accuracy: 0.9098 - loss: 0.2258 - val_accuracy: 0.7778 - val_loss: 0.4379 - lea
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 654ms/step - accuracy: 0.9563 - loss: 0.1818 - val_accuracy: 0.8333 - val_loss: 0.4468 - lea
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 594ms/step - accuracy: 0.9384 - loss: 0.1734 - val_accuracy: 0.8333 - val_loss: 0.4592 - lea
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 703ms/step - accuracy: 0.9678 - loss: 0.1559 - val_accuracy: 0.8333 - val_loss: 0.4571 - lea
```

```
test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 74ms/step - accuracy: 0.7900 - loss: 0.5185
Test Accuracy: 76.32%
```

```
model.save("final_model_tl.h5")

loaded_model = tf.keras.models.load_model("final_model_tl.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
```

```
y_true = test_ds.classes
y_pred_probs = loaded_model.predict(test_ds)
y_pred = np.argmax(y_pred_probs, axis=1)

print("Inference Output: First 20 Samples:")
for i in range(20):
    true_label = class_names[int(y_true[i])]
    pred_label = class_names[int(y_pred[i])]
    print(f"{i+1}. True: {true_label} - Predicted: {pred_label}")
```

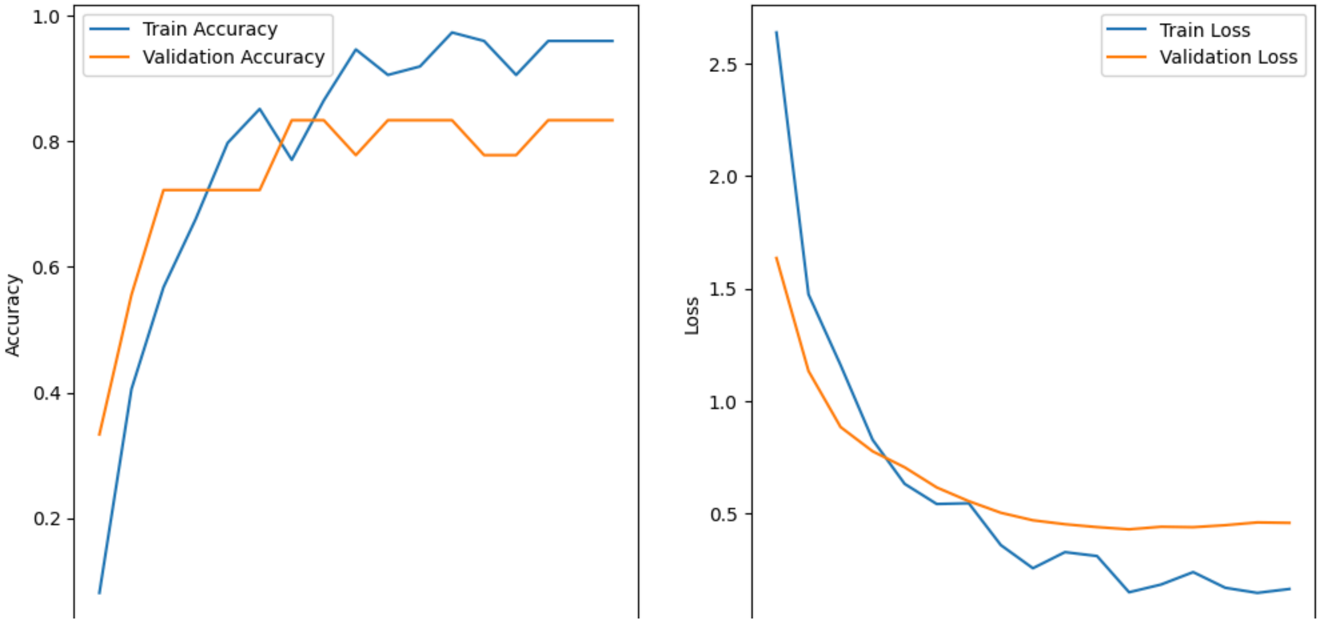
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 2s/step
Inference Output: First 20 Samples:
1. True: acai - Predicted: acai
2. True: acai - Predicted: acai
3. True: acai - Predicted: pupunha
4. True: acai - Predicted: tucuma
5. True: acai - Predicted: acai
6. True: tucuma - Predicted: tucuma
```

```
7. True: tucuma - Predicted: tucuma
8. True: tucuma - Predicted: tucuma
9. True: tucuma - Predicted: tucuma
10. True: tucuma - Predicted: tucuma
11. True: pupunha - Predicted: pupunha
12. True: pupunha - Predicted: pupunha
13. True: pupunha - Predicted: pupunha
14. True: pupunha - Predicted: pupunha
15. True: pupunha - Predicted: pupunha
16. True: guarana - Predicted: guarana
17. True: guarana - Predicted: guarana
18. True: guarana - Predicted: guarana
19. True: guarana - Predicted: guarana
20. True: guarana - Predicted: guarana
```

```
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=class_names))
```

```
Classification Report:
              precision    recall  f1-score   support

        acai       0.60      0.60      0.60         5
      tucuma       0.83      1.00      0.91         5
     pupunha       0.83      1.00      0.91         5
     guarana       0.71      1.00      0.83         5
     cupuacu       0.71      0.62      0.67         8
    graviola       0.86      0.60      0.71        10

    accuracy                           0.76        38
   macro avg       0.76      0.80      0.77        38
weighted avg       0.77      0.76      0.75        38
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```


Train Accuracy
Validation Accuracy
Accuracy
Train Loss
Validation Loss
Loss