PLANT LEAFES DISEASE PREDICTION

DEEP LEARNING

MINI – PROJECT

CODES & OUTPUTS

Group Members :

2020/E/015 - ATPUTHARAVI R.

2020/E/050 - HEMAKANTH N.

2020/E/076 - KUGASARAN S.

Models Used :

Model 01 : CNN – Convolutional Neural Network

Model 02 : RNN – Recurrent Neural Network

CNN Model

Codes :

Training & Validation

Image preprocessing

```
import tensorflow as tf
import numpy as np
import pandas as pd
import seaborn as sns
```

Training & Validation

\+ Code   \+ Markdown

```
import tensorflow as tf
from tensorflow.keras.utils import image_dataset_from_directory

# Path to your dataset directory
dataset_directory = 'Plant disease detection  Dataset'
train_directroy ='Plant disease detection  Dataset/TRAIN'
valid_directory = 'Plant disease detection  Dataset/VALID'

# Load the training dataset
train_dataset = image_dataset_from_directory(
    train_directroy,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=None,
    validation_split=None,  # data used for validation/test

    interpolation="bilinear",
    follow_links=False,
```

```
)

# Load the validation dataset
val_dataset = image_dataset_from_directory(
    valid_directory,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=42,
    validation_split=None,#
    interpolation="bilinear",
    follow_links=False,


)
```

Output :

```
Found 410 files belonging to 2 classes.
Found 355 files belonging to 2 classes.
```

```python
for x,y in train_dataset :
    print(x,x.shape)
    print(y,y.shape)
    break
```

```
tf.Tensor(
[[[[116.71484   96.71484   89.71484 ]
   [124.577896 104.577896  97.577896]
   [126.34276  106.34276   99.34276 ]
   ...
   [149.97382  121.2328   115.10332 ]
   [148.54515  119.54515  113.54515 ]
   [148.78125  118.37789  113.079575]]

  [[132.5653   112.565315 105.565315]
   [134.20634  112.57158  106.1165  ]
   [127.59004  104.59004   98.59004 ]
   ...
   [141.51192  116.32832  109.32832 ]
   [142.48044  115.48044  108.48044 ]
   [142.1684   115.168396 108.168396]]

  [[129.33652  106.33652  100.33652 ]
   [120.41837   97.41837   91.41837 ]
   [116.45804   93.45804   87.45804 ]
   ...
   [138.27737  113.27737  106.27737 ]
   [132.24353  107.24353  100.24353 ]
   [136.72968  111.72968  104.72968 ]]
 ...
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]], shape=(32, 2), dtype=float32) (32, 2)
```

```python
def resize_image(image, label):
    image = tf.image.resize(image, (256, 256))
    return image, label

train_dataset = train_dataset.map(resize_image)
val_dataset = val_dataset.map(resize_image)
```

Building Model

```
from tensorflow.keras.layers import Dense,Conv2D,MaxPool2D,Flatten,Dropout
from tensorflow.keras.models import Sequential
```

```
model = Sequential()
```

```
model.add(Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape=[256,256,3]))
model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
c:\Users\hema2\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an `input_sh
  super().__init__(
```

```
model.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu',input_shape=[256,256,3]))
model.add(Conv2D(filters=64,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu',input_shape=[256,256,3]))
model.add(Conv2D(filters=128,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu',input_shape=[256,256,3]))
model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
model.add(Dropout(0.25)) #to avoid overfitting
```

```
model.add(Flatten())
```

```
model.add(Dense(units=1024,activation='relu'))
```

```
model.add(Dropout(0.40)) #to avoid overfitting
```

Output layer

```
model.add(Dense(units=1,activation='sigmoid',))
```

Compiling

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

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
conv2d_1 (Conv2D)	(None, 254, 254, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_2 (Conv2D)	(None, 127, 127, 64)	18,496
conv2d_3 (Conv2D)	(None, 125, 125, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_4 (Conv2D)	(None, 62, 62, 128)	73,856
conv2d_5 (Conv2D)	(None, 60, 60, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_6 (Conv2D)	(None, 30, 30, 256)	295,168
conv2d_7 (Conv2D)	(None, 28, 28, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
dropout (Dropout)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 1024)	51,381,248
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1)	1,025

```
Total params: 52,554,529 (200.48 MB)

Trainable params: 52,554,529 (200.48 MB)

Non-trainable params: 0 (0.00 B)
```

```python
model.add(Dense(units=2, activation='softmax'))
```

```python
y_pred = model.predict(train_dataset)
predicted_catg = tf.argmax(y_pred, axis=1).numpy()
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 79s 5s/step
```

```python
true_catg = tf.concat([y for x, y in train_dataset], axis=0).numpy()
true_catg = tf.argmax(true_catg, axis=1).numpy()
```

Model Training

```python
# Train the model
history = model.fit(
    train_dataset,
    epochs=5,
    validation_data=val_dataset)
```

```
Epoch 1/5
13/13 ━━━━━━━━━━━━━━━━━━━━ 597s 45s/step - accuracy: 0.8638 - loss: 0.4281 - val_accuracy: 0.9268 - val_loss: 0.3217
Epoch 2/5
13/13 ━━━━━━━━━━━━━━━━━━━━ 545s 41s/step - accuracy: 0.8787 - loss: 0.3886 - val_accuracy: 0.9268 - val_loss: 0.3164
Epoch 3/5
13/13 ━━━━━━━━━━━━━━━━━━━━ 519s 41s/step - accuracy: 0.8666 - loss: 0.4031 - val_accuracy: 0.9268 - val_loss: 0.3117
Epoch 4/5
13/13 ━━━━━━━━━━━━━━━━━━━━ 460s 35s/step - accuracy: 0.8981 - loss: 0.3538 - val_accuracy: 0.9268 - val_loss: 0.3070
Epoch 5/5
13/13 ━━━━━━━━━━━━━━━━━━━━ 462s 35s/step - accuracy: 0.8707 - loss: 0.3927 - val_accuracy: 0.9268 - val_loss: 0.3036
```

To the above when there loss is increasing from 1st to next step

1.Chose small learning rate (Default :0.001 eg :- change it to 0.0001)

2.There may be underfitting , so increase the number of neurons

3.Add more convolution layer to extract more feachture from images.

```python
model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu',

input_shape=[256,256,3]))

model.add(Conv2D(filters=512,kernel_size=3,padding='same'activation='relu'))

model.add(MaxPool2D(pool_size=2,strides=2))
```

Model Evalution

Model Evaluation

```
[23] train_loss,train_acc =model.evaluate(train_dataset)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 42s 3s/step - accuracy: 0.8649 - loss: 0.4010
```

```
[24] print(train_loss)
```

```
0.36488819122314453
```

```
[25] val_loss,val_accuracy =model.evaluate(val_dataset)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 31s 2s/step - accuracy: 0.9323 - loss: 0.2949
```

```
[26] print(val_loss)
```

```
0.30355706810951233
```

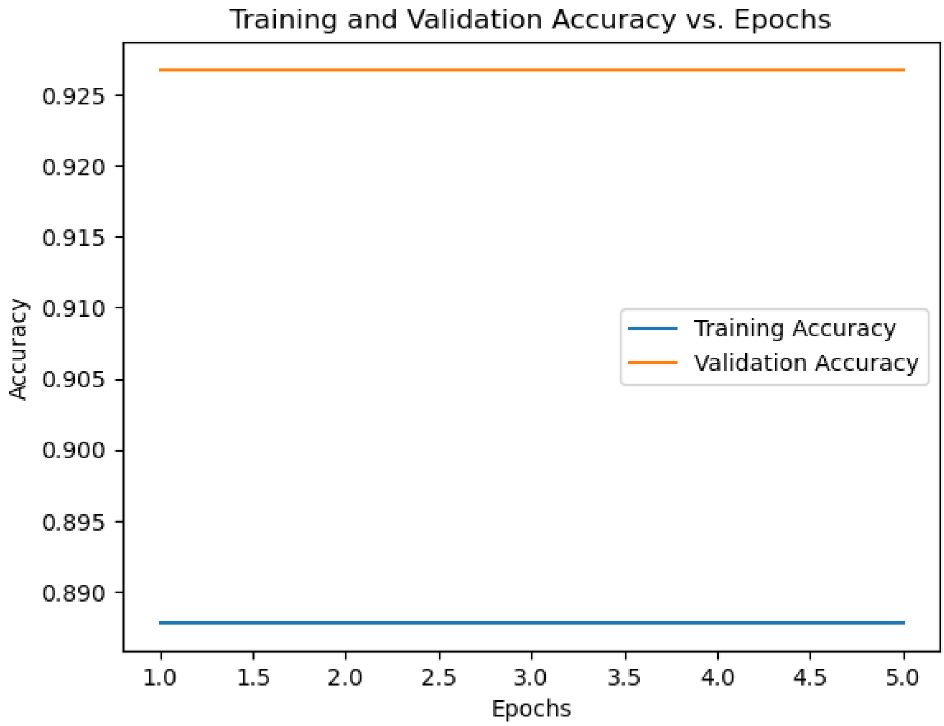
```
[27] history.history   ##uncomment to print training history
```

```
{'accuracy': [0.8878048658370972,
  0.8878048658370972,
  0.8878048658370972,
  0.8878048658370972,
  0.8878048658370972],
 'loss': [0.38396719098091125,
  0.37482762336730957,
  0.37138935923576355,
  0.36888575553894043,
  0.3658979535102844],
 'val_accuracy': [0.9267605543136597,
  0.9267605543136597,
  0.9267605543136597,
  0.9267605543136597,
  0.9267605543136597],
 'val_loss': [0.3216615319252014,
  0.3164416551589966,
  0.31168046593666077,
  0.306998610496521,
  0.3035570979118347]}
```

Acuracy Visualization

```
import numpy as np
x = np.array([1])  # x has shape (1,)
y = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])  # y has shape (10,)

# Reshape x to match the shape of y
x = np.full_like(y, x)  # x now has shape (10,)

# Now you can perform operations on x and y
result = x + y
```

```
import matplotlib.pyplot as plt


train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

# Create a figure and axis object
fig, ax = plt.subplots()

# Plot the training accuracy vs. epochs
ax.plot(range(1, len(train_acc) + 1), train_acc, label='Training Accuracy')

# Plot the validation accuracy vs. epochs
ax.plot(range(1, len(val_acc) + 1), val_acc, label='Validation Accuracy')

# Set the title and labels
ax.set_title('Training and Validation Accuracy vs. Epochs')
ax.set_xlabel('Epochs')
ax.set_ylabel('Accuracy')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```



Other model evaluations

```
test_directory = 'Plant disease detection  Dataset/TEST'
test_dataset = image_dataset_from_directory(
    test_directory,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=42,
    validation_split=None,
    interpolation="bilinear",
    follow_links=False,


)
```

```
Found 410 files belonging to 2 classes.
```

```
y_pred =model.predict(test_dataset)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 53s 3s/step
```

```python
y_pred,y_pred.shape
```

[34]

```
(array([[0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
 ...
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258],
        [0.82840735, 0.17159258]], dtype=float32),
 (410, 2))
```

```python
predicted_catg = tf.cast(tf.greater(y_pred[:, 1], 0.5), tf.bool)
print(predicted_catg)
```

36]

```
tf.Tensor(
[False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
```

```python
true_catg =tf.concat([y for x,y in test_dataset],axis=0)
true_catg
```

[37]

```
<tf.Tensor: shape=(410, 2), dtype=float32, numpy=
array([[0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
```

```python
y_true = tf.cast(tf.greater(true_catg[:, 1], 0.5), tf.bool)
y_true
```

[38]

```
<tf.Tensor: shape=(410,), dtype=bool, numpy=
array([ True, False, False,  True, False, False, False, False, False,
       False, False,  True, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False,  True,
        True, False, False,  True, False, False,  True, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False,  True, False, False, False, False,  True, False, False,
        True,  True, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False,  True, False, False, False,  True, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
```

```python
import os
class_name=test_dataset.class_names
class_name
```

```
['Diseased', 'Healthy']
```

```python
from sklearn.metrics import classification_report
print(classification_report (y_true,predicted_catg,target_names=class_name))
```

```
              precision    recall  f1-score   support

    Diseased       0.89      1.00      0.94       364
     Healthy       0.00      0.00      0.00        46

    accuracy                           0.89       410
   macro avg       0.44      0.50      0.47       410
weighted avg       0.79      0.89      0.84       410
```

```python
for i in range(83):
    print(all_labels[np.argmax(y_test[i])], "-",all_labels[np.argmax(y_pred[i])])
```

```
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Healthy - Diseased
Diseased - Diseased
Healthy - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Healthy - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
Diseased - Diseased
...
Diseased - Diseased
Diseased - Diseased
Healthy - Diseased
Diseased - Diseased
```

```python
from sklearn.metrics import classification_report,confusion_matrix
cn= confusion_matrix (y_true,predicted_catg)
cn.shape
```

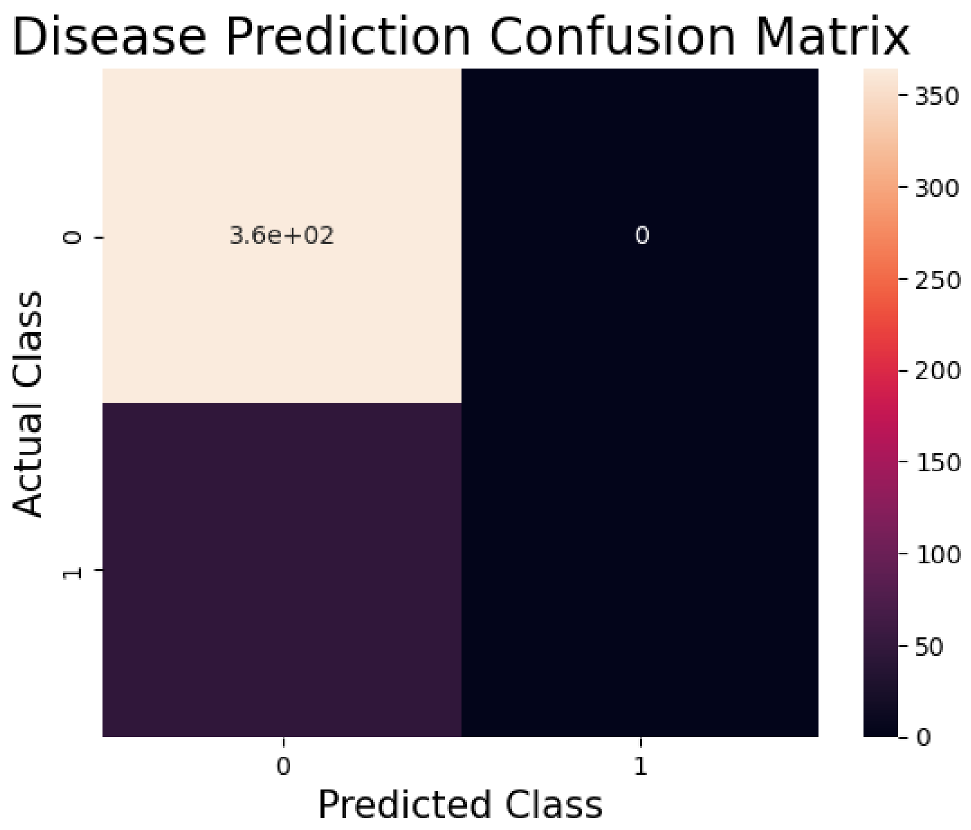
[41]

```
(2, 2)
```

Confusion matrix visuals

```python
import matplotlib.pyplot as plt
sns.heatmap(cn,annot=True)
plt.xlabel("Predicted Class",fontsize=15)
plt.ylabel("Actual Class",fontsize=15)
plt.title("Disease Prediction Confusion Matrix",fontsize=20)
plt.show()
```

[42]




```python
model.save("trained_model_plant_disease111.h5")
model.save("trained_model_111.keras")#for moreoption: search in web tf save model
```

[43]

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)
```

CNN : Prediction

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as  plt
```

```python
model= tf.keras.models.load_model('trained_model_111.keras')
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
conv2d_1 (Conv2D)	(None, 254, 254, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_2 (Conv2D)	(None, 127, 127, 64)	18,496
conv2d_3 (Conv2D)	(None, 125, 125, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_4 (Conv2D)	(None, 62, 62, 128)	73,856
conv2d_5 (Conv2D)	(None, 60, 60, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_6 (Conv2D)	(None, 30, 30, 256)	295,168
conv2d_7 (Conv2D)	(None, 28, 28, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
dropout (Dropout)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 1024)	51,381,248
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1)	1,025
dense_2 (Dense)	(None, 2)	4

```
Total params: 105,109,068 (400.96 MB)

Trainable params: 52,554,533 (200.48 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 52,554,535 (200.48 MB)
```

Data Visualization

```
!pip install opencv-python
```

```
Collecting opencv-python
  Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\hema2\anaconda3\lib\site-packages (from opencv-python)
Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.10.0.84
```

```
import cv2
image_path ="Plant disease detection  Dataset\TEST\Diseased\1013.jpeg"

#Read image
img=cv2.imread(image_path)
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)   #converting BGR to RGB


#Display image
plt.imshow(img)
plt.tittle("Test Image")
plt.xticks([])
plt.ytricks([])
plt.show
```

```
##Testing Model
image =tf.keras.preprocessing.image.load_img(image_path,target_size
=(256,256))
input_arr =tf.keras.preprocessing.image.img_to_array(image)
input_arr =np.array([input_arr])    # converting sigle image to a batch
print(input_arr.shape)
```

```
prediction =model.predict(input_arr)
prediction,prediction.shape
```

```
result_index =np.argmax(prediction)
result_index
class_name =['Healthy','Diseased']
```

```
#Displaying the Result of Test image
model_prediction =class_name[result_index]
plt.imshow(img)
plt.tittle(f"Prediction :{model_prediction}")
plt.xticks([])
plt.ytricks([])
plt.show()
```

Model 2 :RNN

```
# -*- coding: utf-8 -*-
"""Untitled15.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1j7lgHxYK0VOswFJHyFs96cmFvaFmIzIi
"""

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

!ls /content/drive/MyDrive/DL_project/Plant-disease-detection-Dataset

# Commented out IPython magic to ensure Python compatibility.
# %cd
# %pwd /content/drive/MyDrive/DL_project
import os
os.listdir(".")

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Flatten, TimeDistributed
from tensorflow.keras.optimizers import Adam

train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
```

```python
train_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/DL_project/Plant-disease-detection-Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/DL_project/Plant-disease-detection-Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation')

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

model = Sequential()
model.add(cnn_base)
model.add(Flatten())

model.compile(optimizer=Adam(lr=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // 32,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // 32,
    epochs=10)

loss, accuracy = model.evaluate(validation_generator)
print(f'Validation Accuracy: {accuracy*100:.2f}%')
```

Outputs :

```
Epoch 1/20
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `s
  self._warn_if_super_not_called()
10/10 311s 26s/step - accuracy: 0.7266 - loss: 0.5742 - val_accuracy: 0.8438 - val_loss: 0.4616 - learning_rate: 1.0000e-04
Epoch 2/20
 1/10 3:13 22s/step - accuracy: 0.9375 - loss: 0.1920/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data; interrupting training. Mak
  self.gen.throw(typ, value, traceback)
10/10 33s 1s/step - accuracy: 0.9375 - loss: 0.1920 - val_accuracy: 0.7647 - val_loss: 0.5216 - learning_rate: 1.0000e-04
Epoch 3/20
10/10 230s 23s/step - accuracy: 0.9057 - loss: 0.2186 - val_accuracy: 0.6562 - val_loss: 0.6416 - learning_rate: 1.0000e-04
Epoch 4/20
10/10 30s 1s/step - accuracy: 0.9062 - loss: 0.1782 - val_accuracy: 0.7059 - val_loss: 0.5396 - learning_rate: 1.0000e-04
Epoch 5/20
10/10 248s 25s/step - accuracy: 0.9115 - loss: 0.1755 - val_accuracy: 0.7344 - val_loss: 0.5034 - learning_rate: 2.0000e-05
Epoch 6/20
10/10 16s 1s/step - accuracy: 0.8889 - loss: 0.2434 - val_accuracy: 0.8235 - val_loss: 0.4544 - learning_rate: 2.0000e-05
Epoch 7/20
10/10 227s 22s/step - accuracy: 0.9132 - loss: 0.1803 - val_accuracy: 0.7656 - val_loss: 0.4528 - learning_rate: 2.0000e-05
Epoch 8/20
10/10 29s 1s/step - accuracy: 0.9062 - loss: 0.2181 - val_accuracy: 0.7647 - val_loss: 0.4702 - learning_rate: 2.0000e-05
Epoch 9/20
10/10 223s 22s/step - accuracy: 0.9380 - loss: 0.1312 - val_accuracy: 0.5312 - val_loss: 0.8126 - learning_rate: 2.0000e-05
Epoch 10/20
10/10 30s 1s/step - accuracy: 0.9062 - loss: 0.1650 - val_accuracy: 0.6471 - val_loss: 0.5898 - learning_rate: 2.0000e-05
Epoch 11/20
10/10 232s 23s/step - accuracy: 0.9246 - loss: 0.1684 - val_accuracy: 0.5625 - val_loss: 0.8105 - learning_rate: 1.0000e-05
Epoch 12/20
10/10 30s 1s/step - accuracy: 0.9062 - loss: 0.1525 - val_accuracy: 0.4706 - val_loss: 0.7808 - learning_rate: 1.0000e-05
3/3 49s 15s/step - accuracy: 0.8315 - loss: 0.4074
Validation Accuracy: 82.72%
```