Plant_Leaves_Disease_Prediction

March 10, 2024

Deep Learning Practical Assignment 3 A - Plant Leaves Disease Prediction

Name : Ilhe Uddhav Sampat | Batch : B3| Roll no. : 4159

[ ]:

```
 
```

[3]:

```
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).
```

From original plant dataset tomato directory selected for training and testing with only 200 images from each disease. There are 3 diseases and 1 healthy. Total 4 directories selected each for training and testing so total 1600 images considered.

[ ]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪load_img,img_to_array
import numpy as np
```

Dataset is stored on google drive with Train and Test folder.

[ ]:

```
train_dir = r'/content/drive/MyDrive/Colab_Notebooks/3/A/Train'
test_dir = r'/content/drive/MyDrive/Colab_Notebooks/3/A/Test'
```

[ ]:

```
img_size = 224
batch_size = 32
```

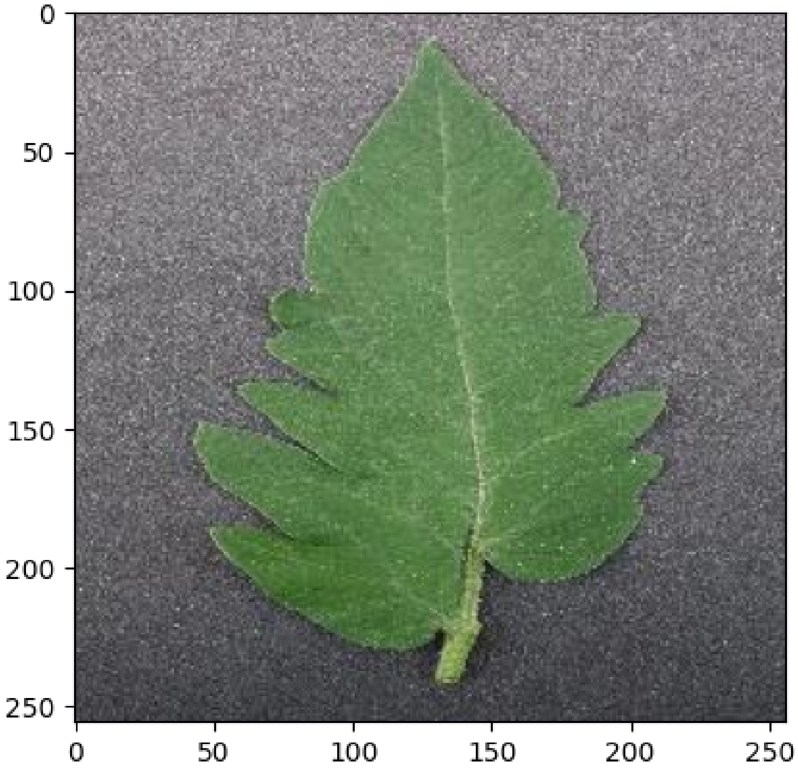
[ ]:

```
# importing required libraries
import matplotlib.pyplot as plt
import matplotlib.image as img

# reading the image
testImage = img.imread('/content/drive/MyDrive/Colab_Notebooks/3/A/Train/
 ↪Tomato___healthy/000bf685-b305-408b-91f4-37030f8e62db___GH_HL Leaf 308.1.
 ↪JPG')

# displaying the image
plt.imshow(testImage)
```

```
[ ]: <matplotlib.image.AxesImage at 0x7b6cf9fa2410>
```



```
[ ]: # importing required libraries
     import matplotlib.pyplot as plt
     import matplotlib.image as img

     # reading the image
     testImage = img.imread('/content/drive/MyDrive/Colab_Notebooks/3/A/Train/
      ↪Tomato___Target_Spot/c6a1dc1f-e0fd-40df-8726-5db2b0be7150___Com.G_TgS_FL␣
      ↪0736.JPG')

     # displaying the image
     plt.imshow(testImage)
```

```
[ ]: <matplotlib.image.AxesImage at 0x7b6cf437ca60>
```

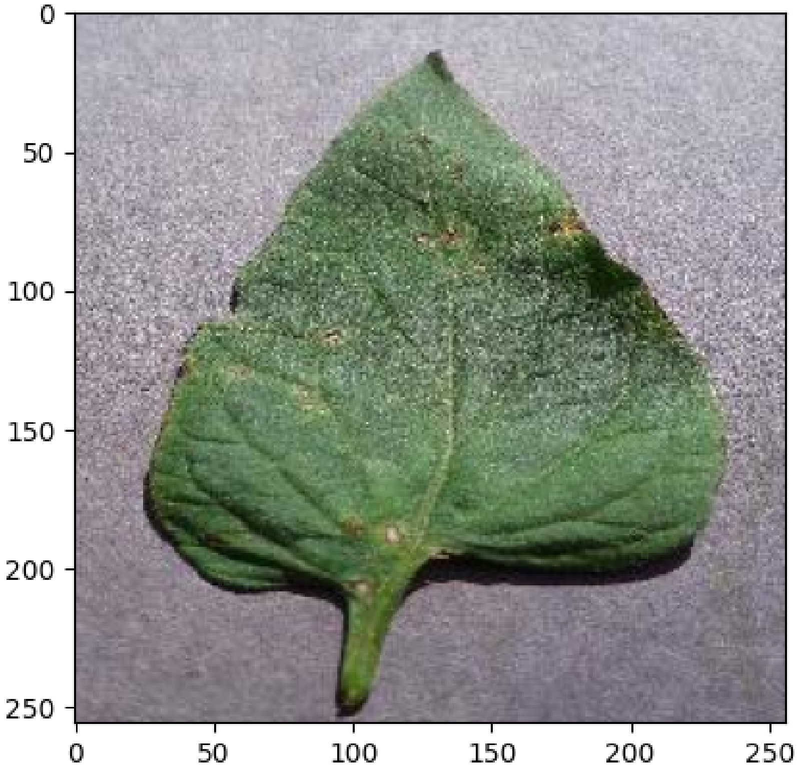


Image Preprocessing

```
[ ]: train_datagen = ImageDataGenerator(rescale=1./255)
     train_generator = train_datagen.
       ↪flow_from_directory(train_dir,target_size=(img_size,img_size),batch_size=batch_size,class_m
```

```
Found 800 images belonging to 4 classes.
```

```
[ ]: test_datagen = ImageDataGenerator(rescale=1./255)
     test_generator = test_datagen.
       ↪flow_from_directory(test_dir,target_size=(img_size,img_size),batch_size=batch_size,class_mo
```

```
Found 800 images belonging to 4 classes.
```

Categories are identified.

```
[ ]: class_names=list(train_generator.class_indices)
     class_names
```

```
[ ]: ['Tomato___Target_Spot',
      'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
      'Tomato___Tomato_mosaic_virus',
      'Tomato___healthy']
```

[ ]:
```
class_names=list(test_generator.class_indices)
class_names
```

[ ]:
```
['Tomato___Target_Spot',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato___Tomato_mosaic_virus',
 'Tomato___healthy']
```

Model Building

[ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,␣
 ↪Dense,Dropout, BatchNormalization
```

[ ]:
```
model = Sequential()
model.add((Conv2D(32, (3,3), activation='relu', input_shape=(img_size,img_size,␣
 ↪3))))
model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))
model.add((Conv2D(64, (3,3), activation='relu')))
model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))
model.add((Conv2D(64, (3,3), activation='relu')))
model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))
model.add((Conv2D(128, (3,3), activation='relu')))
model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))
model.add((Flatten()))
model.add((Dense(128, activation='relu')))
model.add((Dropout(0.2)))
model.add((Dense(64, activation='relu')))
model.add((Dense(train_generator.num_classes, activation='softmax')))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 batch_normalization (Batch  (None, 222, 222, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 111, 111, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496
 batch_normalization_1 (Bat  (None, 109, 109, 64)      256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 54, 54, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 64)        36928

 batch_normalization_2 (Bat  (None, 52, 52, 64)        256
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 26, 26, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 24, 24, 128)       73856

 batch_normalization_3 (Bat  (None, 24, 24, 128)       512
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 12, 12, 128)       0
 g2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 128)               2359424

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 4)                 260

=================================================================
Total params: 2499268 (9.53 MB)
Trainable params: 2498692 (9.53 MB)
Non-trainable params: 576 (2.25 KB)
_________________________________________________________________
```

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

```
[ ]: model.fit(train_generator, epochs=50, validation_data=test_generator)
```

```
Epoch 1/50
25/25 [==============================] - 86s 4s/step - loss: 0.2873 - accuracy:
0.9325 - val_loss: 5.2645 - val_accuracy: 0.2500
Epoch 2/50
```

```
25/25 [==============================] - 7s 293ms/step - loss: 0.2720 -
accuracy: 0.9425 - val_loss: 9.3248 - val_accuracy: 0.2500
Epoch 3/50
25/25 [==============================] - 8s 329ms/step - loss: 0.3011 -
accuracy: 0.9438 - val_loss: 15.0623 - val_accuracy: 0.2500
Epoch 4/50
25/25 [==============================] - 9s 359ms/step - loss: 0.1616 -
accuracy: 0.9638 - val_loss: 17.6249 - val_accuracy: 0.2500
Epoch 5/50
25/25 [==============================] - 7s 291ms/step - loss: 0.2288 -
accuracy: 0.9588 - val_loss: 23.6822 - val_accuracy: 0.2500
Epoch 6/50
25/25 [==============================] - 8s 313ms/step - loss: 0.2553 -
accuracy: 0.9650 - val_loss: 9.7743 - val_accuracy: 0.3575
Epoch 7/50
25/25 [==============================] - 7s 296ms/step - loss: 0.2294 -
accuracy: 0.9563 - val_loss: 8.3915 - val_accuracy: 0.2637
Epoch 8/50
25/25 [==============================] - 8s 321ms/step - loss: 0.2226 -
accuracy: 0.9688 - val_loss: 7.3704 - val_accuracy: 0.2725
Epoch 9/50
25/25 [==============================] - 8s 320ms/step - loss: 0.2396 -
accuracy: 0.9638 - val_loss: 7.8291 - val_accuracy: 0.4775
Epoch 10/50
25/25 [==============================] - 7s 295ms/step - loss: 0.2184 -
accuracy: 0.9663 - val_loss: 14.8919 - val_accuracy: 0.3050
Epoch 11/50
25/25 [==============================] - 8s 329ms/step - loss: 0.1251 -
accuracy: 0.9712 - val_loss: 8.1868 - val_accuracy: 0.3537
Epoch 12/50
25/25 [==============================] - 9s 360ms/step - loss: 0.0632 -
accuracy: 0.9850 - val_loss: 8.9599 - val_accuracy: 0.4863
Epoch 13/50
25/25 [==============================] - 7s 296ms/step - loss: 0.0710 -
accuracy: 0.9850 - val_loss: 6.0895 - val_accuracy: 0.5462
Epoch 14/50
25/25 [==============================] - 8s 315ms/step - loss: 0.0791 -
accuracy: 0.9800 - val_loss: 17.2595 - val_accuracy: 0.3988
Epoch 15/50
25/25 [==============================] - 9s 363ms/step - loss: 0.0972 -
accuracy: 0.9825 - val_loss: 19.1377 - val_accuracy: 0.5362
Epoch 16/50
25/25 [==============================] - 7s 296ms/step - loss: 0.1088 -
accuracy: 0.9775 - val_loss: 23.3534 - val_accuracy: 0.4950
Epoch 17/50
25/25 [==============================] - 8s 304ms/step - loss: 0.1105 -
accuracy: 0.9787 - val_loss: 12.9446 - val_accuracy: 0.6112
Epoch 18/50
```

```
25/25 [==============================] - 8s 310ms/step - loss: 0.2824 -
accuracy: 0.9550 - val_loss: 4.5245 - val_accuracy: 0.6988
Epoch 19/50
25/25 [==============================] - 8s 304ms/step - loss: 0.2114 -
accuracy: 0.9712 - val_loss: 14.5543 - val_accuracy: 0.4663
Epoch 20/50
25/25 [==============================] - 8s 319ms/step - loss: 0.1981 -
accuracy: 0.9675 - val_loss: 16.9442 - val_accuracy: 0.5125
Epoch 21/50
25/25 [==============================] - 7s 286ms/step - loss: 0.3391 -
accuracy: 0.9613 - val_loss: 6.2863 - val_accuracy: 0.6675
Epoch 22/50
25/25 [==============================] - 8s 317ms/step - loss: 0.1595 -
accuracy: 0.9737 - val_loss: 0.5328 - val_accuracy: 0.9550
Epoch 23/50
25/25 [==============================] - 8s 312ms/step - loss: 0.1277 -
accuracy: 0.9775 - val_loss: 7.8561 - val_accuracy: 0.7412
Epoch 24/50
25/25 [==============================] - 7s 295ms/step - loss: 0.2061 -
accuracy: 0.9750 - val_loss: 8.4606 - val_accuracy: 0.5200
Epoch 25/50
25/25 [==============================] - 8s 326ms/step - loss: 0.1436 -
accuracy: 0.9837 - val_loss: 6.9878 - val_accuracy: 0.6100
Epoch 26/50
25/25 [==============================] - 7s 295ms/step - loss: 0.0290 -
accuracy: 0.9950 - val_loss: 7.0252 - val_accuracy: 0.5825
Epoch 27/50
25/25 [==============================] - 8s 323ms/step - loss: 0.0058 -
accuracy: 0.9962 - val_loss: 2.8066 - val_accuracy: 0.7375
Epoch 28/50
25/25 [==============================] - 8s 327ms/step - loss: 0.0129 -
accuracy: 0.9975 - val_loss: 6.8358 - val_accuracy: 0.6175
Epoch 29/50
25/25 [==============================] - 9s 361ms/step - loss: 0.0869 -
accuracy: 0.9912 - val_loss: 6.7311 - val_accuracy: 0.6587
Epoch 30/50
25/25 [==============================] - 7s 297ms/step - loss: 0.0317 -
accuracy: 0.9937 - val_loss: 5.4170 - val_accuracy: 0.6737
Epoch 31/50
25/25 [==============================] - 8s 327ms/step - loss: 0.0723 -
accuracy: 0.9912 - val_loss: 6.3575 - val_accuracy: 0.7013
Epoch 32/50
25/25 [==============================] - 7s 293ms/step - loss: 0.2196 -
accuracy: 0.9737 - val_loss: 6.3610 - val_accuracy: 0.7850
Epoch 33/50
25/25 [==============================] - 10s 390ms/step - loss: 0.0835 -
accuracy: 0.9837 - val_loss: 11.1792 - val_accuracy: 0.6600
Epoch 34/50
```

```
25/25 [==============================] - 9s 361ms/step - loss: 0.1949 -
accuracy: 0.9762 - val_loss: 8.3436 - val_accuracy: 0.7337
Epoch 35/50
25/25 [==============================] - 7s 301ms/step - loss: 0.0695 -
accuracy: 0.9850 - val_loss: 14.3143 - val_accuracy: 0.5775
Epoch 36/50
25/25 [==============================] - 7s 298ms/step - loss: 0.1098 -
accuracy: 0.9862 - val_loss: 2.7282 - val_accuracy: 0.8050
Epoch 37/50
25/25 [==============================] - 8s 311ms/step - loss: 0.0557 -
accuracy: 0.9912 - val_loss: 3.0408 - val_accuracy: 0.8263
Epoch 38/50
25/25 [==============================] - 9s 354ms/step - loss: 0.0089 -
accuracy: 0.9987 - val_loss: 0.7374 - val_accuracy: 0.9688
Epoch 39/50
25/25 [==============================] - 7s 303ms/step - loss: 0.0154 -
accuracy: 0.9950 - val_loss: 0.5602 - val_accuracy: 0.9488
Epoch 40/50
25/25 [==============================] - 7s 298ms/step - loss: 0.0308 -
accuracy: 0.9937 - val_loss: 1.2889 - val_accuracy: 0.9375
Epoch 41/50
25/25 [==============================] - 8s 315ms/step - loss: 0.0019 -
accuracy: 1.0000 - val_loss: 1.4797 - val_accuracy: 0.9337
Epoch 42/50
25/25 [==============================] - 7s 282ms/step - loss: 0.0084 -
accuracy: 0.9987 - val_loss: 1.3584 - val_accuracy: 0.9438
Epoch 43/50
25/25 [==============================] - 7s 297ms/step - loss: 0.0148 -
accuracy: 0.9962 - val_loss: 1.0129 - val_accuracy: 0.9613
Epoch 44/50
25/25 [==============================] - 8s 321ms/step - loss: 0.0086 -
accuracy: 0.9975 - val_loss: 1.0448 - val_accuracy: 0.9488
Epoch 45/50
25/25 [==============================] - 8s 309ms/step - loss: 0.0265 -
accuracy: 0.9975 - val_loss: 0.8521 - val_accuracy: 0.9675
Epoch 46/50
25/25 [==============================] - 7s 287ms/step - loss: 0.0117 -
accuracy: 0.9962 - val_loss: 0.7557 - val_accuracy: 0.9513
Epoch 47/50
25/25 [==============================] - 8s 322ms/step - loss: 0.0060 -
accuracy: 0.9987 - val_loss: 1.0277 - val_accuracy: 0.9262
Epoch 48/50
25/25 [==============================] - 8s 305ms/step - loss: 0.0113 -
accuracy: 0.9962 - val_loss: 0.5763 - val_accuracy: 0.9650
Epoch 49/50
25/25 [==============================] - 7s 290ms/step - loss: 0.0307 -
accuracy: 0.9925 - val_loss: 0.5787 - val_accuracy: 0.9600
Epoch 50/50
```

```
25/25 [==============================] - 8s 311ms/step - loss: 0.0284 -
accuracy: 0.9950 - val_loss: 0.6495 - val_accuracy: 0.9650
```

```
[ ]: <keras.src.callbacks.History at 0x7b6cf41ed8a0>
```

```
[ ]: loss, accuracy = model.evaluate(test_generator)
     print("Loss :",loss)
     print("Accuracy (Test Data) :",accuracy*100)
```

```
25/25 [==============================] - 4s 143ms/step - loss: 0.6495 -
accuracy: 0.9650
Loss : 0.6494652032852173
Accuracy (Test Data) : 96.49999737739563
```

Image is selected and predicted.

```
[ ]: img_path =r'/content/drive/MyDrive/Colab_Notebooks/3/A/Train/
       ↪Tomato___Target_Spot/c6a1dc1f-e0fd-40df-8726-5db2b0be7150___Com.G_TgS_FL␣
       ↪0736.JPG'
     img = load_img(img_path, target_size=(224, 224))
     img_array = img_to_array(img)
     img_array = np.expand_dims(img_array, axis=0)
     img_array /= 255
```

```
[ ]: print(img_array.shape)
```

```
(1, 224, 224, 3)
```

```
[ ]: prediction = model.predict(img_array)
```

```
1/1 [==============================] - 0s 18ms/step
```

```
[ ]: predicted_class = np.argmax(prediction)
     print('Predicted class:', class_names[predicted_class])
```

```
Predicted class: Tomato___Target_Spot
```

```
[ ]: img_path =r'/content/drive/MyDrive/Colab_Notebooks/3/A/Train/
       ↪Tomato___Tomato_mosaic_virus/dcb74f2b-c523-4147-b9ce-690800411273___PSU_CG␣
       ↪2154_270deg.JPG'
     img = load_img(img_path, target_size=(224, 224))
     img_array = img_to_array(img)
     img_array = np.expand_dims(img_array, axis=0)
     img_array /= 255
     prediction = model.predict(img_array)
     predicted_class = np.argmax(prediction)
     print('Predicted class:', class_names[predicted_class])
```

```
1/1 [==============================] - 0s 18ms/step
Predicted class: Tomato___Tomato_mosaic_virus
```

```
[ ]: img_path =r'/content/drive/MyDrive/Colab_Notebooks/3/A/Test/
      ↪Tomato___Tomato_Yellow_Leaf_Curl_Virus/
      ↪bf09ead6-7015-4942-bbf7-e509193885ab___YLCV_NREC 2830.JPG'
     img = load_img(img_path, target_size=(224, 224))
     img_array = img_to_array(img)
     img_array = np.expand_dims(img_array, axis=0)
     img_array /= 255
     prediction = model.predict(img_array)
     predicted_class = np.argmax(prediction)
     print('Predicted class:', class_names[predicted_class])
```

```
1/1 [==============================] - 0s 18ms/step
Predicted class: Tomato___Tomato_Yellow_Leaf_Curl_Virus
```