**Implementation of a CNN model to identify the disease of plant leaf, using Python in jupyter notebook.**

**Student Name: Gargi Darade.**

**Subject/Class: CMPS 451 Artificial Intelligence.**

#Import the required libraries

import os

import matplotlib.image as mpimg

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras import layers, models

import numpy as np

from PIL import Image

#current\_path = os.getcwd()

#print(current\_path)

Number of Classes.

# Dataset Path

base\_dir = 'plantvillage'

print("Number of Classes:"+str(len(os.listdir(base\_dir))))

img\_classes=os.listdir(base\_dir)

print("Image classes:")

print(img\_classes)

Number of Classes:15

Image classes:

['Pepper\_\_bell\_\_\_Bacterial\_spot', 'Pepper\_\_bell\_\_\_healthy', 'Potato\_\_\_Early\_blight', 'Potato\_\_\_healthy', 'Potato\_\_\_Late\_blight', 'Tomato\_Bacterial\_spot', 'Tomato\_Early\_blight', 'Tomato\_healthy', 'Tomato\_Late\_blight', 'Tomato\_Leaf\_Mold', 'Tomato\_Septoria\_leaf\_spot', 'Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite', 'Tomato\_\_Target\_Spot', 'Tomato\_\_Tomato\_mosaic\_virus', 'Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus']

**Sample images from all classes**

flCnt=1

for img\_class in img\_classes:

print("Image class:"+img\_class)

tmp\_file=os.listdir(base\_dir+"/"+img\_class)[:1]

image\_path =base\_dir+"/"+img\_class+"/"+ tmp\_file[0]

print("Image path:"+image\_path)

# Read the image

img = mpimg.imread(image\_path)

print("Image shape:"+str(img.shape))

# Display the image

plt.imshow(img)

plt.axis('off') # Turn off axis numbers

plt.savefig("clsRepImg" + str(flCnt) +".png")

flCnt=flCnt+1

plt.show()

Image class:Pepper\_\_bell\_\_\_Bacterial\_spot

Image path:plantvillage/Pepper\_\_bell\_\_\_Bacterial\_spot/0022d6b7-d47c-4ee2-ae9a-392a53f48647\_\_\_JR\_B.Spot 8964.JPG

Image shape:(256, 256, 3)



Image class:Pepper\_\_bell\_\_\_healthy

Image path:plantvillage/Pepper\_\_bell\_\_\_healthy/00100ffa-095e-4881-aebf-61fe5af7226e\_\_\_JR\_HL 7886.JPG

Image shape:(256, 256, 3)



Image class:Potato\_\_\_Early\_blight

Image path:plantvillage/Potato\_\_\_Early\_blight/001187a0-57ab-4329-baff-e7246a9edeb0\_\_\_RS\_Early.B 8178.JPG

Image shape:(256, 256, 3)



Image class:Potato\_\_\_healthy

Image path:plantvillage/Potato\_\_\_healthy/00fc2ee5-729f-4757-8aeb-65c3355874f2\_\_\_RS\_HL 1864.JPG

Image shape:(256, 256, 3)



Image class:Potato\_\_\_Late\_blight

Image path:plantvillage/Potato\_\_\_Late\_blight/0051e5e8-d1c4-4a84-bf3a-a426cdad6285\_\_\_RS\_LB 4640.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Bacterial\_spot

Image path:plantvillage/Tomato\_Bacterial\_spot/00416648-be6e-4bd4-bc8d-82f43f8a7240\_\_\_GCREC\_Bact.Sp 3110.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Early\_blight

Image path:plantvillage/Tomato\_Early\_blight/0012b9d2-2130-4a06-a834-b1f3af34f57e\_\_\_RS\_Erly.B 8389.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_healthy

Image path:plantvillage/Tomato\_healthy/000146ff-92a4-4db6-90ad-8fce2ae4fddd\_\_\_GH\_HL Leaf 259.1.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Late\_blight

Image path:plantvillage/Tomato\_Late\_blight/0003faa8-4b27-4c65-bf42-6d9e352ca1a5\_\_\_RS\_Late.B 4946.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Leaf\_Mold

Image path:plantvillage/Tomato\_Leaf\_Mold/00694db7-3327-45e0-b4da-a8bb7ab6a4b7\_\_\_Crnl\_L.Mold 6923.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Septoria\_leaf\_spot

Image path:plantvillage/Tomato\_Septoria\_leaf\_spot/002533c1-722b-44e5-9d2e-91f7747b2543\_\_\_Keller.St\_CG 1831.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite

Image path:plantvillage/Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite/002835d1-c18e-4471-aa6e-8d8c29585e9b\_\_\_Com.G\_SpM\_FL 8584.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_\_Target\_Spot

Image path:plantvillage/Tomato\_\_Target\_Spot/002213fb-b620-4593-b9ac-6a6cc119b100\_\_\_Com.G\_TgS\_FL 8360.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_\_Tomato\_mosaic\_virus

Image path:plantvillage/Tomato\_\_Tomato\_mosaic\_virus/000ec6ea-9063-4c33-8abe-d58ca8a88878\_\_\_PSU\_CG 2169.JPG

Image shape:(256, 256, 3)



Image class:Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus

Image path:plantvillage/Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus/00139ae8-d881-4edb-925f-46584b0bd68c\_\_\_YLCV\_NREC 2944.JPG

Image shape:(256, 256, 3)



# Image Parameters

img\_size = 224

batch\_size = 32

**Train Test Split**

# Image Data Generators

data\_gen = ImageDataGenerator(

rescale=1./255,

validation\_split=0.2 # Use 20% of data for validation

)

# Train Generator

train\_generator = data\_gen.flow\_from\_directory(

base\_dir,

target\_size=(img\_size, img\_size),

batch\_size=batch\_size,

subset='training',

class\_mode='categorical'

)

Found 16516 images belonging to 15 classes.

# Validation Generator

validation\_generator = data\_gen.flow\_from\_directory(

base\_dir,

target\_size=(img\_size, img\_size),

batch\_size=batch\_size,

subset='validation',

class\_mode='categorical'

)

Found 4122 images belonging to 15 classes.

### Convolutional Neural Network

# Model Definition

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_size, img\_size, 3)))

model.add(layers.MaxPooling2D(2, 2))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D(2, 2))

model.add(layers.Flatten())

model.add(layers.Dense(256, activation='relu'))

model.add(layers.Dense(train\_generator.num\_classes, activation='softmax'))

# model summary

model.summary()

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_2 (Conv2D) (None, 222, 222, 32) 896

max\_pooling2d\_2 (MaxPooling (None, 111, 111, 32) 0

2D)

conv2d\_3 (Conv2D) (None, 109, 109, 64) 18496

max\_pooling2d\_3 (MaxPooling (None, 54, 54, 64) 0

2D)

flatten\_1 (Flatten) (None, 186624) 0

dense\_2 (Dense) (None, 256) 47776000

dense\_3 (Dense) (None, 15) 3855

=================================================================

Total params: 47,799,247

Trainable params: 47,799,247

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Model training

# Training the Model

history = model.fit(

train\_generator,

steps\_per\_epoch=train\_generator.samples // batch\_size, # Number of steps per epoch

epochs=10, # Number of epochs

validation\_data=validation\_generator,

validation\_steps=validation\_generator.samples // batch\_size # Validation steps

)

Epoch 1/10

516/516 [==============================] - 585s 1s/step - loss: 1.1255 - accuracy: 0.6690 - val\_loss: 0.5227 - val\_accuracy: 0.8262

Epoch 2/10

516/516 [==============================] - 583s 1s/step - loss: 0.3572 - accuracy: 0.8829 - val\_loss: 0.4483 - val\_accuracy: 0.8452

Epoch 3/10

516/516 [==============================] - 584s 1s/step - loss: 0.1372 - accuracy: 0.9547 - val\_loss: 0.4470 - val\_accuracy: 0.8613

Epoch 4/10

516/516 [==============================] - 583s 1s/step - loss: 0.0767 - accuracy: 0.9756 - val\_loss: 0.5408 - val\_accuracy: 0.8660

Epoch 5/10

516/516 [==============================] - 583s 1s/step - loss: 0.0619 - accuracy: 0.9811 - val\_loss: 0.6105 - val\_accuracy: 0.8401

Epoch 6/10

516/516 [==============================] - 583s 1s/step - loss: 0.0478 - accuracy: 0.9849 - val\_loss: 0.6552 - val\_accuracy: 0.8430

Epoch 7/10

516/516 [==============================] - 583s 1s/step - loss: 0.0568 - accuracy: 0.9817 - val\_loss: 1.0894 - val\_accuracy: 0.7869

Epoch 8/10

516/516 [==============================] - 582s 1s/step - loss: 0.0331 - accuracy: 0.9888 - val\_loss: 0.7412 - val\_accuracy: 0.8433

Epoch 9/10

516/516 [==============================] - 582s 1s/step - loss: 0.0471 - accuracy: 0.9864 - val\_loss: 0.7312 - val\_accuracy: 0.8552

Epoch 10/10

516/516 [==============================] - 582s 1s/step - loss: 0.0207 - accuracy: 0.9930 - val\_loss: 0.7479 - val\_accuracy: 0.8633

#### Model Evaluation

# Model Evaluation

print("Evaluating model...")

val\_loss, val\_accuracy = model.evaluate(validation\_generator, steps=validation\_generator.samples // batch\_size)

print(f"Validation Accuracy: {val\_accuracy \* 100:.2f}%")

Evaluating model...

128/128 [==============================] - 27s 211ms/step - loss: 0.7465 - accuracy: 0.8643

Validation Accuracy: 86.43%

# Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.savefig("ModelAccuracy.png")

plt.show()

# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

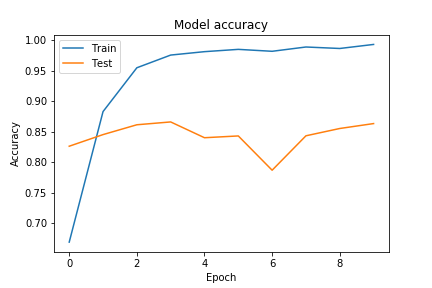
plt.ylabel('Loss')

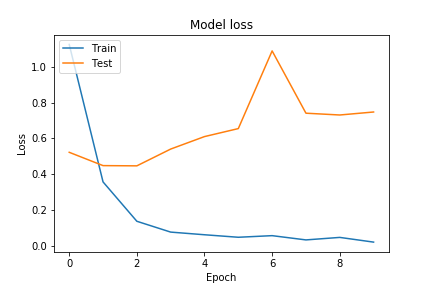
plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.savefig("ModelLoss.png")

plt.show()





#### Building a Predictive System

# Function to Load and Preprocess the Image using Pillow

def load\_and\_preprocess\_image(image\_path, target\_size=(224, 224)):

# Load the image

img = Image.open(image\_path)

# Resize the image

img = img.resize(target\_size)

# Convert the image to a numpy array

img\_array = np.array(img)

# Add batch dimension

img\_array = np.expand\_dims(img\_array, axis=0)

# Scale the image values to [0, 1]

img\_array = img\_array.astype('float32') / 255.

return img\_array

# Function to Predict the Class of an Image

def predict\_image\_class(model, image\_path, class\_indices):

preprocessed\_img = load\_and\_preprocess\_image(image\_path)

predictions = model.predict(preprocessed\_img)

predicted\_class\_index = np.argmax(predictions, axis=1)[0]

predicted\_class\_name = class\_indices[predicted\_class\_index]

return predicted\_class\_name

# Create a mapping from class indices to class names

class\_indices = {v: k for k, v in train\_generator.class\_indices.items()}

class\_indices

{0: 'Pepper\_\_bell\_\_\_Bacterial\_spot',

1: 'Pepper\_\_bell\_\_\_healthy',

2: 'Potato\_\_\_Early\_blight',

3: 'Potato\_\_\_Late\_blight',

4: 'Potato\_\_\_healthy',

5: 'Tomato\_Bacterial\_spot',

6: 'Tomato\_Early\_blight',

7: 'Tomato\_Late\_blight',

8: 'Tomato\_Leaf\_Mold',

9: 'Tomato\_Septoria\_leaf\_spot',

10: 'Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite',

11: 'Tomato\_\_Target\_Spot',

12: 'Tomato\_\_Tomato\_YellowLeaf\_\_Curl\_Virus',

13: 'Tomato\_\_Tomato\_mosaic\_virus',

14: 'Tomato\_healthy'}

# Example Usage

image\_path = 'test\_images/test\_potato\_early\_blight.jpg'

predicted\_class\_name = predict\_image\_class(model, image\_path, class\_indices)

# Output the result

print("Predicted Class Name:", predicted\_class\_name)

1/1 [==============================] - 0s 83ms/step

Predicted Class Name: Potato\_\_\_Early\_blight

#### Save the model to local drive

model.save('PlantDiseasePredictionCNNModel.h5')