Potato leaf disease detection using CNN

A Project Report
Submitted in Partial fulfillment of Major Project for the award of Bachelor of Technology in (Computer Science & Engineering)

Submitted to
ITM UNIVERSITY GWALIOR (M.P.)

Minor PROJECT REPORT
Subject Code: CSD0602[P]

Submitted by
Aditya Goyal [BETN1CS19078]
Aman Gupta [BETN1CS19079]
Shivam Shukla [BETN1CS19099]

Under the supervision of

Asst. Prof Vani Agarwal

Undertaken At

School of Engineering & Technology
ITM University, Gwalior

Dept. of Computer Science & Applications
ITM UNIVERSITY, GWALIOR (M.P.)
SESSION Jan-June 2022

UNIVERSITY
GWALIOR • MP • INDIA
“CELEBRATING DREAMS”

CERTIFICATE

This is to Certify that, Aditya goyal, Aman Gupta, Shivam Shukla students of B.Tech VI Semester , January– June 2022 session of this school has completed his VI semester project entitled News Recommendation Portal (Assure News).

He has submitted a satisfactory project report for the award of degree of Bachelor of Technology in (Computer Science & Engineering) of ITM University, Gwalior.

Dr. Shashikant Gupta	**Mrs. Vani Agrawal**
Associate Professor & Head,	Asst. Professor
Dept. of Computer Science & Applications	Dept. of Computer Science & Applications
ITM University, Gwalior	ITM University, Gwalior

Acknowledgment

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning of the people where ceaseless cooperation made it possible, whose constant guidance and encouragement crown all the efforts with success.

We shall always be grateful to our parent whose moral support at every step of our life is the reason to us of using our self. Their encouragement in every joy and believe of life laid the real foundation for our personality.

We sincerely thank to our project coordinator **Ms. Neha Garg** (Assistant Professor, Dept. of Computer Science & Applications, ITM UNIVERSITY) for their great help, guidance, support and ideas towards our project. We are highly grateful to –

Dr. Shashikant Gupta (HOD, Dept. of Computer Science & Applications, ITM UNIVERSITY)

Miss Vani Agrawal (Asst.Prof, Dept. of Computer Science & Applications, ITM UNIVERSITY)

for providing us their valuable guidance, suggestions, encouragement & ideas during whole project work.

We are also thankful to staff members for providing us the technical support to carry out the project work and guidance at each & every step during the project work.

We are also very thankful to lab assistants for their co-operation. It was their careful evaluation, keen interest, timely guidance and valuable suggestions that steered us out of difficulties at every stage of the project. It was great pleasure and good learning experience to have worked under their guidance during tenure of the project.

We also convey our sincere thanks to all our friends who ardently encouraged us through their precious advices and helping attitude in all the fields.

Aditya Goyal - BETN1CS19078
Aman Gupta - BETN1CS19079
Shivam Shukla – BETN1CS19099

Introduction

We developed an End-to-End project which is based on Pure Deep Learning.

The reason behind building this project is to detect or identify potato leaf diseases, having a variety of illnesses. Because our naked eyes can’t classify them, but Convolutional Neural Network (CNN) can easily.

Problem Statement

- Farmers who grow potatoes suffer from serious financial losses each year which cause several diseases that affect potato plants.

- Potato leaf disease detection in an early stage is challenging because of variations in crop species, crop diseases symptoms and environmental factors.These factor make it difficult to detect potato leaf diseases in the early stage.

Types of diseases :-

- Bacterial wilt.
- Septoria leaf spot.
- Late blight.
- Early blight.
- Common scab.
- Black scurf/canker.

The most frequent diseases in potato leaf are: -

Early blight : -

Early blight is caused by fungus.

Late blight : -

Late blight is caused by specific micro-organisms.

Project Description

- We used Convolutional Neural Network - Deep Learning to diagnose plant diseases.

- We developed an end-to-end Deep Learning project in the field of agriculture. We created a simple Image Classification Model that will categorize Potato Leaf Disease using a simple and classic Convolutional Neural Network Architecture.

This project can break down into some steps:

- Data Collection
- Data Cleaning & Data Preprocessing
- Model building
- Model deployment

Data Collection

Any Data Science project starts with the process of acquiring the data. First, we need to collect data. We have 3 options for collecting data

- we can use readymade data we can either buy it from a third-party vendor or get it from Kaggle etc.

- we can have a team of Data Anatator whose job is to collect these images from farmers and annotate those images either healthy potato leaves or having early or late blight diseases. So this team of annotators works with farmers, go to the fields and they can ask farmers to take a photograph of leaves or they can take photographs themselves and they can classify them with the help of experts from agriculture field. So they can manually collect the data. But this process will be time-consuming.

- writing a web-scraping script to go through different websites which has potato images and collect those images and use different tools to annotate the data.
- In this project, I am using readymade data that I got from Kaggle and ATLIQ (Agriculture)

In this dataset we have three types of images:

Healty Leaf

Early Bright

Late Blight

Data Cleaning & Data Preprocessing

Pre-processing data aims to improve the quality of data to realize an accurate training model output. The first step is to minimize the noise in the image, and if there is excessive noise in the image, then it will not be used. Acquired images have a variety of sizes, and images are resized to Same pixels to standardize the input of images in datasets

Model building

Import all the Dependencies

In [1]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

Set all the Constants

In [2]:
```
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=50
```

Import data into tensorflow dataset object

In [3]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 2152 files belonging to 3 classes.
```

In [4]:
```
class_names = dataset.class_names
class_names
```

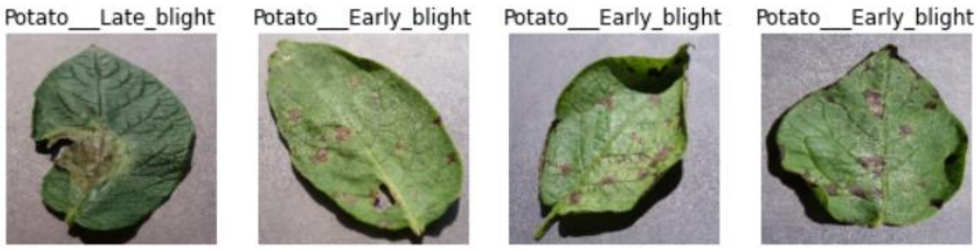
Out[4]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [5]:
```
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
```

```
(32, 256, 256, 3)
[1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 1 2 0 0]
```

Visualize some of the images from our dataset

In [6]:
```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



In [7]:
```
len(dataset)
```
Out[7]: 68

In [8]:
```
train_size = 0.8
len(dataset)*train_size
```
Out[8]: 54.400000000000006

In [9]:
```
train_ds = dataset.take(54)
len(train_ds)
```
Out[9]: 54

In [10]:
```
test_ds = dataset.skip(54)
len(test_ds)
```
Out[10]: 14

In [14]:
```
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
```

In [15]:
```
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

In [16]:
```
len(train_ds)
```
Out[16]: 54

In [17]:
```
len(val_ds)
```
Out[17]: 6

In [18]:
```
len(test_ds)
```
Out[18]: 8

Cache, Shuffle, and Prefetch the Dataset

In [19]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Building the Model

In [20]:
```
resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
])
```

Data Augmentation

In [21]:
```
data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
```

Applying Data Augmentation to Train Dataset

In [22]:
```
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [23]:
```
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

In [24]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0         
                                                                 
 conv2d (Conv2D)             (32, 254, 254, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (32, 127, 127, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (32, 62, 62, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (32, 30, 30, 64)         0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (32, 14, 14, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (32, 6, 6, 64)           0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928     
```

```
 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

 max_pooling2d_5 (MaxPooling  (32, 2, 2, 64)           0
 2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 64)                  16448

 dense_1 (Dense)             (32, 3)                   195

=================================================================
Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0
_________________________________________________________________
```

Compiling the Model

We use `adam` Optimizer, `SparseCategoricalCrossentropy` for losses, `accuracy` as a metric

In [25]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [26]:
```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

```
Epoch 1/50
54/54 [==============================] - 142s 2s/step - loss: 0.9018 - accuracy: 0.4688 - val_loss: 0.8595 - val_accuracy: 0.
96
Epoch 2/50
54/54 [==============================] - 129s 2s/step - loss: 0.7173 - accuracy: 0.6707 - val_loss: 0.6332 - val_accuracy: 0.
44
Epoch 3/50
54/54 [==============================] - 127s 2s/step - loss: 0.5207 - accuracy: 0.7899 - val_loss: 0.6678 - val_accuracy: 0.
83
Epoch 4/50
54/54 [==============================] - 128s 2s/step - loss: 0.3354 - accuracy: 0.8519 - val_loss: 0.2808 - val_accuracy: 0.
02
Epoch 5/50
54/54 [==============================] - 133s 2s/step - loss: 0.2197 - accuracy: 0.9190 - val_loss: 0.2174 - val_accuracy: 0.
67
Epoch 6/50
54/54 [==============================] - 137s 3s/step - loss: 0.1446 - accuracy: 0.9410 - val_loss: 0.1778 - val_accuracy: 0.
75
Epoch 7/50
54/54 [==============================] - 133s 2s/step - loss: 0.1131 - accuracy: 0.9572 - val_loss: 0.4487 - val_accuracy: 0.
94
Epoch 8/50
54/54 [==============================] - 137s 3s/step - loss: 0.0763 - accuracy: 0.9728 - val_loss: 0.1923 - val_accuracy: 0.
79
```

```
Epoch 48/50
54/54 [==============================] - 123s 2s/step - loss: 0.0092 - accuracy: 0.9977 - val_loss: 0.0094 - val_accuracy: 0.9
48
Epoch 49/50
54/54 [==============================] - 121s 2s/step - loss: 0.0146 - accuracy: 0.9948 - val_loss: 0.0148 - val_accuracy: 0.9
48
Epoch 50/50
54/54 [==============================] - 124s 2s/step - loss: 0.0199 - accuracy: 0.9919 - val_loss: 0.0032 - val_accuracy: 1.0
00
```

In [27]:
```
scores = model.evaluate(test_ds)
```

```
8/8 [==============================] - 7s 565ms/step - loss: 0.0597 - accuracy: 0.9844
```

In [28]:
```
scores
```

Out[28]:
```
[0.059664420783519745, 0.984375]
```

Plotting the Accuracy and Loss Curves

In [29]:
```
history
```

Out[29]:
```
<keras.callbacks.History at 0x1aa6a99a580>
```

In [30]:
```
history.params
```

Out[30]:
```
{'verbose': 1, 'epochs': 50, 'steps': 54}
```

In [31]:
```
history.history.keys()
```

Out[31]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [32]:
```
type(history.history['loss'])
```

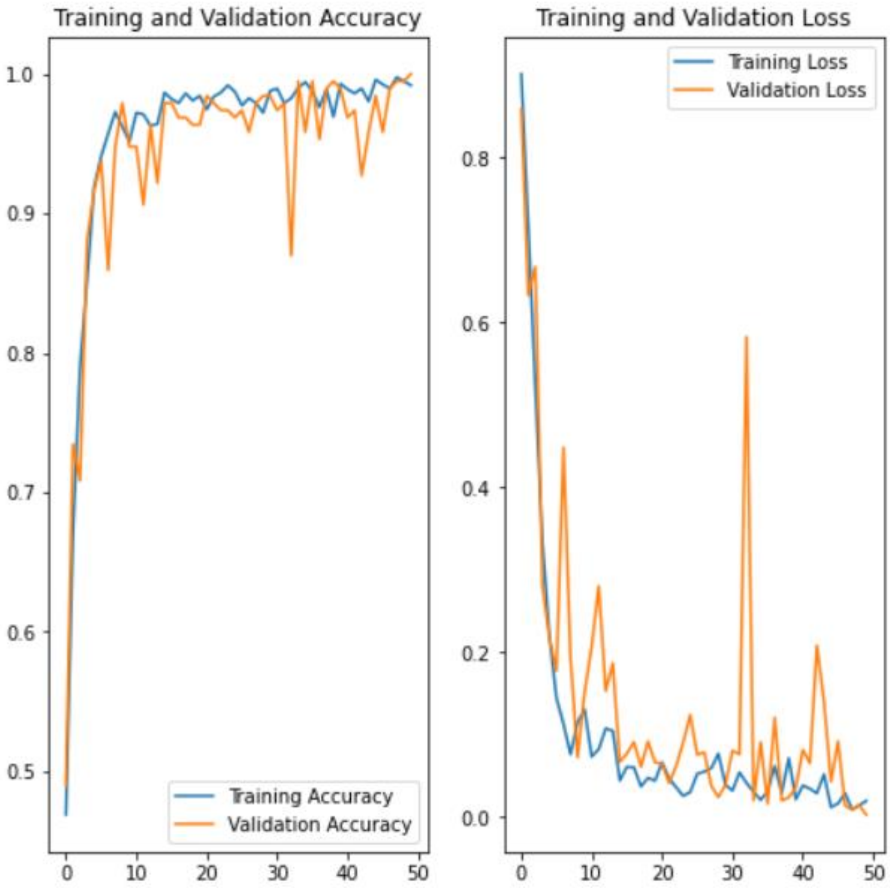
Out[32]:
```
list
```

In [33]:
```
len(history.history['loss'])
```

Out[33]:
```
50
```

In [34]:
```
history.history['loss'][:5] # show loss for first 5 epochs
```

Out[34]:
```
[0.901808500289917,
 0.717258870601654,
 0.5206604599952698,
 0.3354171812534332,
 0.21974077820777893]
```



Run prediction on a sample image

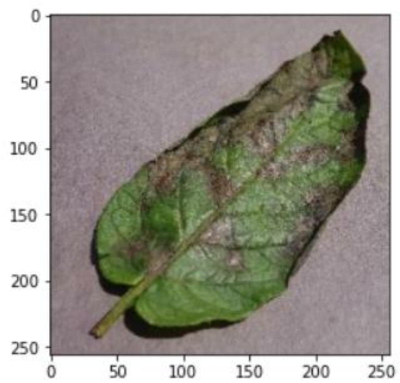
In [49]:
```
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Potato___Late_blight
1/1 [==============================] - 1s 536ms/step
predicted label: Potato___Late_blight
```



Saving the Model

We append the model to the list of models as a new version

In [42]:
```
model.save("../potatoes.h5")
```