plantdisease-train

April 30, 2024

1 Plant Disease Prediction

1.1 Importing Dataset

Dataset Link: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

1.2 Importing libraries

[33]:
```python
import tensorflow as tf
from tensorflow.keras import models, utils, optimizers
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import os
import json
from zipfile import ZipFile
from PIL import Image
from IPython.display import HTML
```

[2]:
```python
kaggle_credentails = json.load(open("kaggle.json"))

os.environ["KAGGLE_USERNAME"] = kaggle_credentails["username"]
os.environ["KAGGLE_KEY"] = kaggle_credentails["key"]

!kaggle datasets download -d shaharshh/plant-disease-detection-dataset

with ZipFile("plant-disease-detection-dataset.zip", 'r') as zip_ref:
    zip_ref.extractall()

!rm  "plant-disease-detection-dataset.zip"
```

```
Downloading new-plant-diseases-dataset.zip to /content
100% 2.69G/2.70G [00:31<00:00, 89.2MB/s]
100% 2.70G/2.70G [00:31<00:00, 91.9MB/s]
```

1.3 Data Preprocessing

1.3.1 Training Image preprocessing

[3]:
```
training_set = utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 70295 files belonging to 38 classes.
```

1.3.2 Validation Image Preprocessing

[4]:
```
validation_set = utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 17572 files belonging to 38 classes.
```

[5]:
```
class_names = training_set.class_names
class_names
```

[5]:

```
['Apple___Apple_scab',
 'Apple___Black_rot',
 'Apple___Cedar_apple_rust',
 'Apple___healthy',
 'Blueberry___healthy',
 'Cherry_(including_sour)___Powdery_mildew',
 'Cherry_(including_sour)___healthy',
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
 'Corn_(maize)___Common_rust_',
 'Corn_(maize)___Northern_Leaf_Blight',
 'Corn_(maize)___healthy',
 'Grape___Black_rot',
 'Grape___Esca_(Black_Measles)',
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
 'Grape___healthy',
 'Orange___Haunglongbing_(Citrus_greening)',
 'Peach___Bacterial_spot',
 'Peach___healthy',
 'Pepper,_bell___Bacterial_spot',
 'Pepper,_bell___healthy',
 'Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Raspberry___healthy',
 'Soybean___healthy',
 'Squash___Powdery_mildew',
 'Strawberry___Leaf_scorch',
 'Strawberry___healthy',
 'Tomato___Bacterial_spot',
 'Tomato___Early_blight',
 'Tomato___Late_blight',
 'Tomato___Leaf_Mold',
 'Tomato___Septoria_leaf_spot',
 'Tomato___Spider_mites Two-spotted_spider_mite',
 'Tomato___Target_Spot',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato___Tomato_mosaic_virus',
 'Tomato___healthy']
```

1.4 Building Model

[7]:

```
cnn = models.Sequential()

cnn.
 ↪add(Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape=[128,128,3
cnn.add(MaxPool2D(pool_size=2,strides=2))
cnn.add(Dropout(0.25))
```

```
cnn.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(MaxPool2D(pool_size=2,strides=2))

cnn.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(MaxPool2D(pool_size=2,strides=2))
cnn.add(Dropout(0.25))

cnn.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(MaxPool2D(pool_size=2,strides=2))

cnn.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))
cnn.add(MaxPool2D(pool_size=2,strides=2))
cnn.add(Dropout(0.25))

cnn.add(Flatten())
cnn.add(Dense(units=1500,activation='relu'))
cnn.add(Dropout(0.4))

cnn.add(Dense(units=38,activation='softmax'))
```

[8]:
```
cnn.compile(optimizer=optimizers.Adam(
    learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
```

[9]:
```
cnn.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 max_pooling2d (MaxPooling2  (None, 64, 64, 32)        0
 D)

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        36928

 max_pooling2d_2 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)
```

```
 dropout_1 (Dropout)         (None, 16, 16, 64)        0

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_3 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 64)          36928

 max_pooling2d_4 (MaxPoolin  (None, 4, 4, 64)          0
 g2D)

 dropout_2 (Dropout)         (None, 4, 4, 64)          0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 1500)              1537500

 dropout_3 (Dropout)         (None, 1500)              0

 dense_1 (Dense)             (None, 38)                57038

=================================================================
Total params: 1724714 (6.58 MB)
Trainable params: 1724714 (6.58 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[10]:

```
training_history = cnn.
 ↪fit(x=training_set,validation_data=validation_set,epochs=10)
```

```
Epoch 1/10
2197/2197 [==============================] - 109s 46ms/step - loss: 3.0623 -
accuracy: 0.2346 - val_loss: 1.9025 - val_accuracy: 0.4936
Epoch 2/10
2197/2197 [==============================] - 98s 44ms/step - loss: 1.2859 -
accuracy: 0.6128 - val_loss: 0.9892 - val_accuracy: 0.7251
Epoch 3/10
2197/2197 [==============================] - 79s 36ms/step - loss: 0.7981 -
accuracy: 0.7539 - val_loss: 0.5564 - val_accuracy: 0.8449
Epoch 4/10
2197/2197 [==============================] - 87s 39ms/step - loss: 0.5710 -
accuracy: 0.8209 - val_loss: 0.4645 - val_accuracy: 0.8573
Epoch 5/10
2197/2197 [==============================] - 93s 42ms/step - loss: 0.4322 -
accuracy: 0.8610 - val_loss: 0.3923 - val_accuracy: 0.8786
Epoch 6/10
2197/2197 [==============================] - 83s 38ms/step - loss: 0.3476 -
```

```
accuracy: 0.8865 - val_loss: 0.2675 - val_accuracy: 0.9187
Epoch 7/10
2197/2197 [==============================] - 77s 35ms/step - loss: 0.2853 -
accuracy: 0.9067 - val_loss: 0.2383 - val_accuracy: 0.9264
Epoch 8/10
2197/2197 [==============================] - 79s 36ms/step - loss: 0.2456 -
accuracy: 0.9186 - val_loss: 0.2320 - val_accuracy: 0.9267
Epoch 9/10
2197/2197 [==============================] - 85s 39ms/step - loss: 0.2130 -
accuracy: 0.9287 - val_loss: 0.1899 - val_accuracy: 0.9391
Epoch 10/10
2197/2197 [==============================] - 87s 40ms/step - loss: 0.1872 -
accuracy: 0.9379 - val_loss: 0.1883 - val_accuracy: 0.9389
```

1.5 Evaluating Model

[11]:
```
# Training Accuracy
train_loss, train_acc = cnn.evaluate(training_set)
print('Training accuracy:', train_acc)
```

```
2197/2197 [==============================] - 55s 25ms/step - loss: 0.1183 -
accuracy: 0.9660
Training accuracy: 0.9659719467163086
```

[12]:
```
# Validation Accuracy
val_loss, val_acc = cnn.evaluate(validation_set)
print('Validation accuracy:', val_acc)
```

```
550/550 [==============================] - 12s 22ms/step - loss: 0.1883 -
accuracy: 0.9389
Validation accuracy: 0.9389369487762451
```

1.5.1 Saving Model

[13]:
```
cnn.save('trained_plant_disease_model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

[14]:
```
training_history.history
```

[14]:
```
{'loss': [3.062305212020874,
  1.28593111038208,
  0.7981175780296326,
  0.5709856152534485,
```

```
  0.4321916103363037,
  0.34760773181915283,
  0.28534531593322754,
  0.24562574923038483,
  0.21301718056201935,
  0.18715494871139526],
 'accuracy': [0.2346397340297699,
  0.6127747297286987,
  0.7538800835609436,
  0.8209260702133179,
  0.8609573841094971,
  0.8865352869033813,
  0.906735897064209,
  0.9186002016067505,
  0.9287146925926208,
  0.9379187822341919],
 'val_loss': [1.9025198221206665,
  0.9891754388809204,
  0.556390106678009,
  0.4644668698310852,
  0.39225131273269653,
  0.2675146162509918,
  0.23834316432476044,
  0.2319595366716385,
  0.18992629647254944,
  0.18827539682388306],
 'val_accuracy': [0.4935693144798279,
  0.7250739932060242,
  0.844923734664917,
  0.8572729229927063,
  0.8785567879676819,
  0.9187343716621399,
  0.9263601303100586,
  0.926701545715332,
  0.9390507340431213,
  0.9389369487762451]}
```

[15]:
```
import json
with open('training_hist.json','w') as f:
  json.dump(training_history.history,f)
```

[16]:
```
print(training_history.history.keys())
```

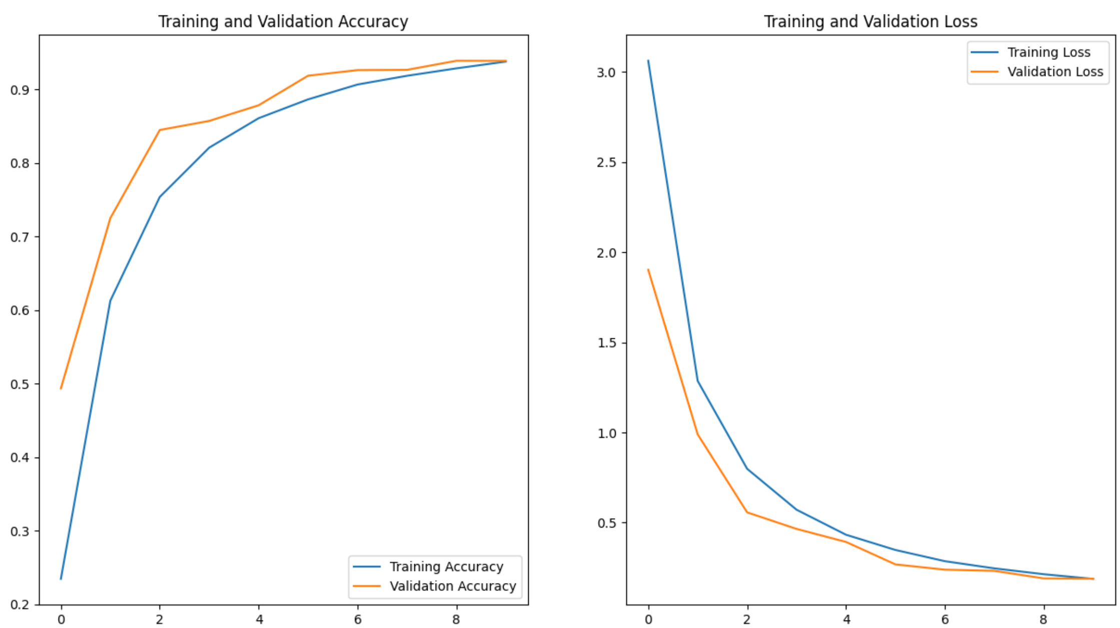
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

1.6 Visualising results

[17]:
```
acc = training_history.history['accuracy']
val_acc = training_history.history['val_accuracy']
loss = training_history.history['loss']
val_loss = training_history.history['val_loss']
```

[18]:
```
plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1)
plt.plot(range(10), acc, label='Training Accuracy')
plt.plot(range(10), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(10), loss, label='Training Loss')
plt.plot(range(10), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



1.7 Model evaluation

[21]:
```
class_name = validation_set.class_names
```

[22]:
```
len(class_name)
```

[22]: 38

[28]:
```
test_set = utils.image_dataset_from_directory(
    'test',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=False,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

Found 17572 files belonging to 38 classes.

[30]:
```
# Predicted

y_pred = cnn.predict(test_set)
predicted_categories = tf.argmax(y_pred, axis=1)
predicted_categories
```

[30]: <tf.Tensor: shape=(17572,), dtype=int64, numpy=array([0, 0, 0, …, 37, 37, 37])>

[32]:
```
# Actual

true_categories = tf.concat([y for x, y in test_set], axis=0)
Y_true = tf.argmax(true_categories, axis=1)
Y_true
```

[32]: <tf.Tensor: shape=(17572,), dtype=int64, numpy=array([0, 0, 0, …, 37, 37, 37])>

[26]:
```
len(Y_true)
```

[26]: 33

[27]:
```
len(predicted_categories)
```

[27]: 33

1.8 Precision Recall Fscore

[36]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[37]:
```
# Precision Recall Fscore
print(classification_report(Y_true, predicted_categories,␣
 ↪target_names=class_name))
```

```
                                                    precision    recall
f1-score   support

                            Apple___Apple_scab       0.97      0.91
0.94       504
                             Apple___Black_rot       0.94      0.99
0.96       497
                      Apple___Cedar_apple_rust       0.89      0.99
0.94       440
                               Apple___healthy       0.91      0.96
0.93       502
                           Blueberry___healthy       0.96      0.94
0.95       454
      Cherry_(including_sour)___Powdery_mildew       1.00      0.96
0.98       421
             Cherry_(including_sour)___healthy       0.95      0.89
0.92       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.89      0.91
0.90       410
                    Corn_(maize)___Common_rust_       0.99      0.99
0.99       477
            Corn_(maize)___Northern_Leaf_Blight       0.93      0.96
0.94       477
                         Corn_(maize)___healthy       0.97      1.00
0.99       465
                             Grape___Black_rot       0.93      0.94
0.94       472
                  Grape___Esca_(Black_Measles)       0.96      0.97
0.97       480
    Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       0.97      0.97
0.97       430
                               Grape___healthy       0.91      0.99
0.95       423
      Orange___Haunglongbing_(Citrus_greening)       0.93      0.98
0.95       503
                        Peach___Bacterial_spot       0.91      0.89
0.90       459
                               Peach___healthy       0.95      0.99
0.97       432
                 Pepper,_bell___Bacterial_spot       0.94      0.93
```

			0.94	478
Pepper,_bell___healthy	0.96	0.84	0.89	497
Potato___Early_blight	0.93	0.98	0.96	485
Potato___Late_blight	0.97	0.84	0.90	485
Potato___healthy	0.95	0.95	0.95	456
Raspberry___healthy	0.94	0.97	0.95	445
Soybean___healthy	0.84	0.99	0.91	505
Squash___Powdery_mildew	0.98	0.97	0.98	434
Strawberry___Leaf_scorch	0.96	0.95	0.96	444
Strawberry___healthy	1.00	0.98	0.99	456
Tomato___Bacterial_spot	0.85	0.95	0.90	425
Tomato___Early_blight	0.85	0.82	0.84	480
Tomato___Late_blight	0.93	0.82	0.87	463
Tomato___Leaf_Mold	0.98	0.93	0.95	470
Tomato___Septoria_leaf_spot	0.93	0.76	0.84	436
Tomato___Spider_mites Two-spotted_spider_mite	0.93	0.95	0.94	435
Tomato___Target_Spot	0.91	0.86	0.88	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.97	0.98	0.97	490
Tomato___Tomato_mosaic_virus	0.99	0.96	0.97	448
Tomato___healthy	0.95	1.00	0.97	481
accuracy			0.94	17572
macro avg	0.94	0.94	0.94	17572
weighted avg	0.94	0.94	0.94	17572

[34]:
```
HTML("""
<img src="https://www.researchgate.net/publication/336402347/figure/fig3/AS:
↪812472659349505@1570719985505/
↪Calculation-of-Precision-Recall-and-Accuracy-in-the-confusion-matrix.ppm" />
""")
```

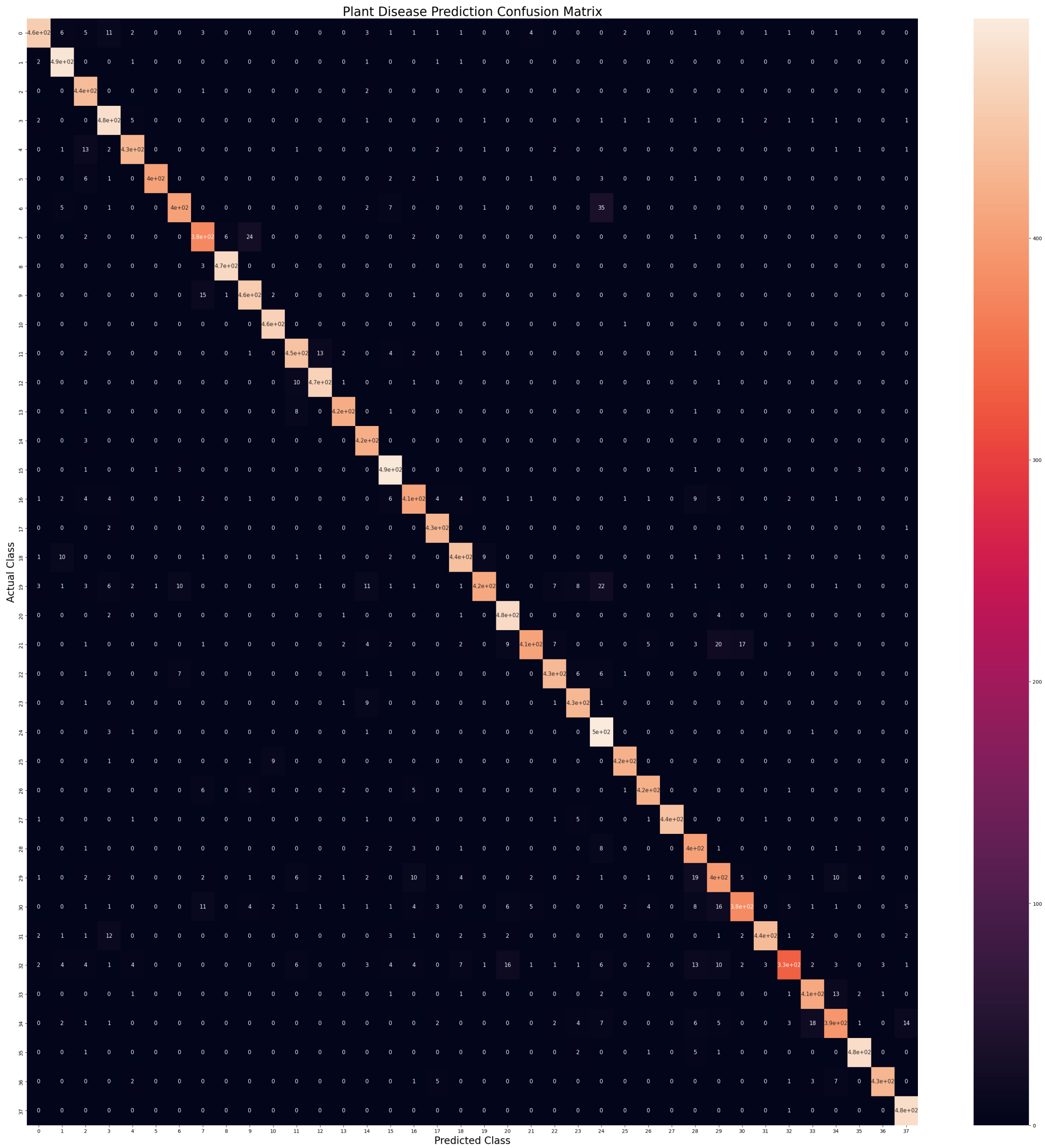
[34]: <IPython.core.display.HTML object>

1.8.1 Confusion Matrix Visualization

[38]:
```
cm = confusion_matrix(Y_true, predicted_categories)
```

[42]:
```
plt.figure(figsize=(40, 40))
sns.heatmap(cm,annot=True,annot_kws={"size": 11})

plt.xlabel('Predicted Class',fontsize = 20)
plt.ylabel('Actual Class',fontsize = 20)
plt.title('Plant Disease Prediction Confusion Matrix',fontsize = 25)
plt.show()
```


Plant Disease Prediction Confusion Matrix
Actual Class
Predicted Class

[29]:

```
```