Double-click (or enter) to edit

```
import zipfile
import os

# Path to your uploaded zip
zip_path = "/content/drive/MyDrive/archive.zip"

# Where to extract the files
extract_path = "/content/plant_dataset"

# Create folder if it doesn't exist
os.makedirs(extract_path, exist_ok=True)

# Extract the zip
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

# Verify files
print("Dataset extracted to:", extract_path)
print("Contents:", os.listdir(extract_path))
```

```
Dataset extracted to: /content/plant_dataset
Contents: ['new plant diseases dataset(augmented)', 'test', 'New Plant Diseases Dataset(Augmented)']
```

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

```
Mounted at /content/drive
```

```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
import tensorflow as tf
import os

# Paths to train and validation folders
base_path = "/content/plant_dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)"
train_dir = os.path.join(base_path, "train")
valid_dir = os.path.join(base_path, "valid")
```

```
training_set = tf.keras.utils.image_dataset_from_directory(
    train_dir,
    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.
```

```
validation_set = tf.keras.utils.image_dataset_from_directory(
    valid_dir,
    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.
```

```
training_set
```

```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 128, 128, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,
38), dtype=tf.float32, name=None))>
```

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

```
 [[[ 53.75  48.75  45.75]
   [ 55.5   50.5   47.5 ]
   [ 56.5   51.5   48.5 ]
   ...
   [120.   110.   108.  ]
   [124.25 114.25 112.25]
   [118.25 108.25 106.25]]

  [[ 59.    54.    51.  ]
   [ 60.5   55.5   52.5 ]
   [ 60.75  55.75  52.75]
   ...
   [118.   108.   106.  ]
   [122.25 112.25 110.25]
   [119.25 109.25 107.25]]

  [[ 60.5   55.5   52.5 ]
   [ 61.    56.    53.  ]
   [ 60.75  55.75  52.75]
   ...
   [124.75 114.75 112.75]
   [121.25 111.25 109.25]
   [111.   101.    99.  ]]

  ...

  [[ 70.75  66.75  67.75]
   [ 72.75  68.75  69.75]
   [ 73.25  69.25  70.25]
   ...
   [158.5  153.5  150.5 ]
   [165.   160.   157.  ]
   [173.5  168.5  165.5 ]]

  [[ 84.25  80.25  81.25]
   [ 84.25  80.25  81.25]
   [ 75.75  71.75  72.75]
   ...
   [152.5  147.5  144.5 ]
   [161.75 156.75 153.75]
   [164.75 159.75 156.75]]

  [[ 72.5   68.5   69.5 ]
   [ 75.75  71.75  72.75]
   [ 73.25  69.25  70.25]
   ...
   [169.   164.   161.  ]
   [166.75 161.75 158.75]
   [173.25 168.25 165.25]]]], shape=(32, 128, 128, 3), dtype=float32) (32, 128, 128, 3)
tf.Tensor(
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]], shape=(32, 38), dtype=float32) (32, 38)
```

```
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=[128,128,3]))
model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shap
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

```
model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=512,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=1500,activation='relu'))
```

```
model.add(Dropout(0.4))
```

```
#Output Layer
model.add(Dense(units=38,activation='softmax'))
```

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
conv2d_1 (Conv2D)	(None, 126, 126, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_2 (Conv2D)	(None, 63, 63, 64)	18,496
conv2d_3 (Conv2D)	(None, 61, 61, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_4 (Conv2D)	(None, 30, 30, 128)	73,856
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_6 (Conv2D)	(None, 14, 14, 256)	295,168
conv2d_7 (Conv2D)	(None, 12, 12, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_8 (Conv2D)	(None, 6, 6, 512)	1,180,160
conv2d_9 (Conv2D)	(None, 4, 4, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1500)	3,073,500
dropout_1 (Dropout)	(None, 1500)	0
dense_1 (Dense)	(None, 38)	57,038

Total params: 7,842,762 (29.92 MB)
Trainable params: 7,842,762 (29.92 MB)
Non-trainable params: 0 (0.00 B)

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

```
Epoch 1/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 164s 67ms/step - accuracy: 0.4047 - loss: 2.1128 - val_accuracy: 0.8370 - val_loss: 0.5148
Epoch 2/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 136s 62ms/step - accuracy: 0.8354 - loss: 0.5269 - val_accuracy: 0.8953 - val_loss: 0.3219
Epoch 3/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 147s 67ms/step - accuracy: 0.9030 - loss: 0.2982 - val_accuracy: 0.9245 - val_loss: 0.2368
Epoch 4/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 138s 63ms/step - accuracy: 0.9349 - loss: 0.1999 - val_accuracy: 0.9304 - val_loss: 0.2270
Epoch 5/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 148s 67ms/step - accuracy: 0.9514 - loss: 0.1485 - val_accuracy: 0.9479 - val_loss: 0.1698
Epoch 6/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 139s 63ms/step - accuracy: 0.9610 - loss: 0.1193 - val_accuracy: 0.9531 - val_loss: 0.1586
Epoch 7/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 137s 62ms/step - accuracy: 0.9688 - loss: 0.0953 - val_accuracy: 0.9611 - val_loss: 0.1266
Epoch 8/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 138s 63ms/step - accuracy: 0.9731 - loss: 0.0820 - val_accuracy: 0.9339 - val_loss: 0.2295
Epoch 9/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 147s 67ms/step - accuracy: 0.9756 - loss: 0.0731 - val_accuracy: 0.9550 - val_loss: 0.1623
Epoch 10/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 147s 67ms/step - accuracy: 0.9801 - loss: 0.0643 - val_accuracy: 0.9595 - val_loss: 0.1349
```

```
#Model Evaluation on Training set
train_loss,train_acc = model.evaluate(training_set)
```

```
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 50s 23ms/step - accuracy: 0.9877 - loss: 0.0381
```

```
print(train_loss,train_acc)
```

```
0.03674689680337906 0.988121509552002
```

```
#Model on Validation set
val_loss,val_acc = model.evaluate(validation_set)
```

```
550/550 ━━━━━━━━━━━━━━━━━━━━ 12s 22ms/step - accuracy: 0.9581 - loss: 0.1443
```

```
print(val_loss,val_acc)
```

```
0.13489656150341034 0.959537923336029
```

```
model.save("trained_model.keras")
```

```
training_history.history
```

```
{'accuracy': [0.6035706400871277,
  0.8583682775497437,
  0.9125969409942627,
  0.9401522278785706,
  0.9542641639709473,
  0.9645351767539978,
  0.9708656668663025,
  0.9742371439933777,
  0.9781208038330078,
  0.9818906188011169],
 'loss': [1.3492861986160278,
  0.448035329580307,
  0.26822665333747864,
  0.184686541557312,
  0.138835147023201,
  0.1092589870095253,
  0.09063984453678131,
  0.07748395204544067,
  0.06538664549589157,
  0.05777793377637863],
 'val_accuracy': [0.8370134234428406,
  0.8952879309654236,
  0.9244821071624756,
  0.9304006099700928,
  0.9478716254234314,
  0.9531072378158569,
  0.9611313343048096,
  0.9338720440864563,
  0.9549852013587952,
  0.959537923336029],
 'val_loss': [0.5148194432258606,
  0.3219480514526367,
  0.23680168390274048,
  0.22698509693145752,
  0.1697731912136078,
  0.1585581749677658,
  0.12656964361667633,
  0.22953073680400848,
  0.16226521134376526,
  0.13489659130573273]}
```

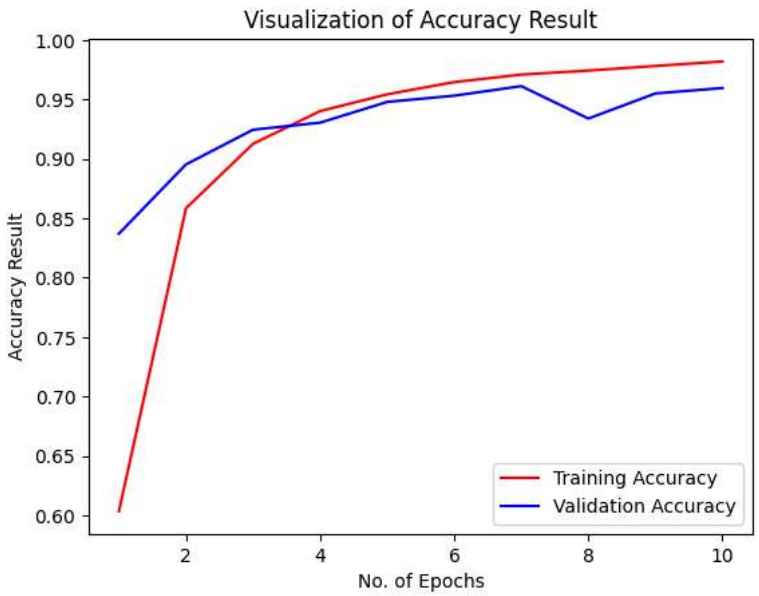
```
#Recording History in json
import json
with open("training_hist.json","w") as f:
    json.dump(training_history.history,f)
```

```
training_history.history['val_accuracy']
```

```
[0.8370134234428406,
 0.8952879309654236,
 0.9244821071624756,
 0.9304006099700928,
 0.9478716254234314,
 0.9531072378158569,
 0.9611313343048096,
 0.9338720440864563,
 0.9549852013587952,
 0.959537923336029]
```

```
epochs = [i for i in range(1,11)]
plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')
plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation Accuracy')
plt.xlabel("No. of Epochs")
plt.ylabel("Accuracy Result")
plt.title("Visualization of Accuracy Result")
```

```
plt.legend()
plt.show()
```




```
class_name = validation_set.class_names
class_name
```

```
['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']
```

```
test_set = tf.keras.utils.image_dataset_from_directory(
    valid_dir,
    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.
```

```
y_pred = model.predict(test_set)
y_pred,y_pred.shape
```

```
550/550 ━━━━━━━━━━━━━━━━━━━━ 14s 24ms/step
(array([[9.9797207e-01, 1.0591141e-05, 1.2627933e-06, ..., 2.6054114e-09,
         4.4939665e-09, 1.9549341e-07],
        [4.4756529e-01, 3.8603536e-04, 2.1126751e-05, ..., 1.5987759e-08,
         1.1897414e-09, 5.2375901e-07],
        [9.9998283e-01, 3.7782092e-06, 8.3672717e-07, ..., 9.8472619e-11,
         7.2623596e-10, 5.4123749e-08],
        ...,
        [8.5626569e-13, 3.8804120e-16, 1.1937019e-11, ..., 6.3979647e-15,
         6.0571246e-15, 1.0000000e+00],
        [1.0717041e-11, 6.5662502e-14, 1.0915025e-10, ..., 5.6728532e-12,
         1.0580083e-12, 9.9999964e-01],
        [1.7173893e-16, 1.5996842e-16, 1.6425755e-15, ..., 8.2035538e-17,
         1.0981301e-15, 1.0000000e+00]], dtype=float32),
 (17572, 38))
```

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

```
predicted_categories
```

```
<tf.Tensor: shape=(17572,), dtype=int64, numpy=array([ 0,  3,  0, ..., 37, 37, 37])>
```

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

```
<tf.Tensor: shape=(17572, 38), dtype=float32, numpy=
array([[1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)>
```

```
Y_true = tf.argmax(true_categories,axis=1)
Y_true
```

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

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(Y_true,predicted_categories,target_names=class_name))
```

```
                                                 precision    recall  f1-score   support

                                Apple___Apple_scab       0.99      0.88      0.93       504
                                 Apple___Black_rot       0.98      0.97      0.97       497
                          Apple___Cedar_apple_rust       0.96      0.98      0.97       440
                                   Apple___healthy       0.90      0.94      0.92       502
                               Blueberry___healthy       0.97      0.98      0.97       454
          Cherry_(including_sour)___Powdery_mildew       0.99      0.98      0.98       421
                 Cherry_(including_sour)___healthy       0.98      0.99      0.99       456
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot       0.98      0.82      0.90       410
                       Corn_(maize)___Common_rust_       1.00      0.99      0.99       477
               Corn_(maize)___Northern_Leaf_Blight       0.88      0.98      0.93       477
                            Corn_(maize)___healthy       0.98      1.00      0.99       465
                                 Grape___Black_rot       0.98      0.99      0.99       472
                      Grape___Esca_(Black_Measles)       0.99      0.98      0.99       480
        Grape___Leaf_blight_(Isariopsis_Leaf_Spot)       0.99      1.00      1.00       430
                                   Grape___healthy       0.99      1.00      1.00       423
          Orange___Haunglongbing_(Citrus_greening)       0.99      0.98      0.98       503
                            Peach___Bacterial_spot       0.97      0.93      0.95       459
                                   Peach___healthy       0.98      0.99      0.99       432
                     Pepper,_bell___Bacterial_spot       0.95      0.97      0.96       478
```

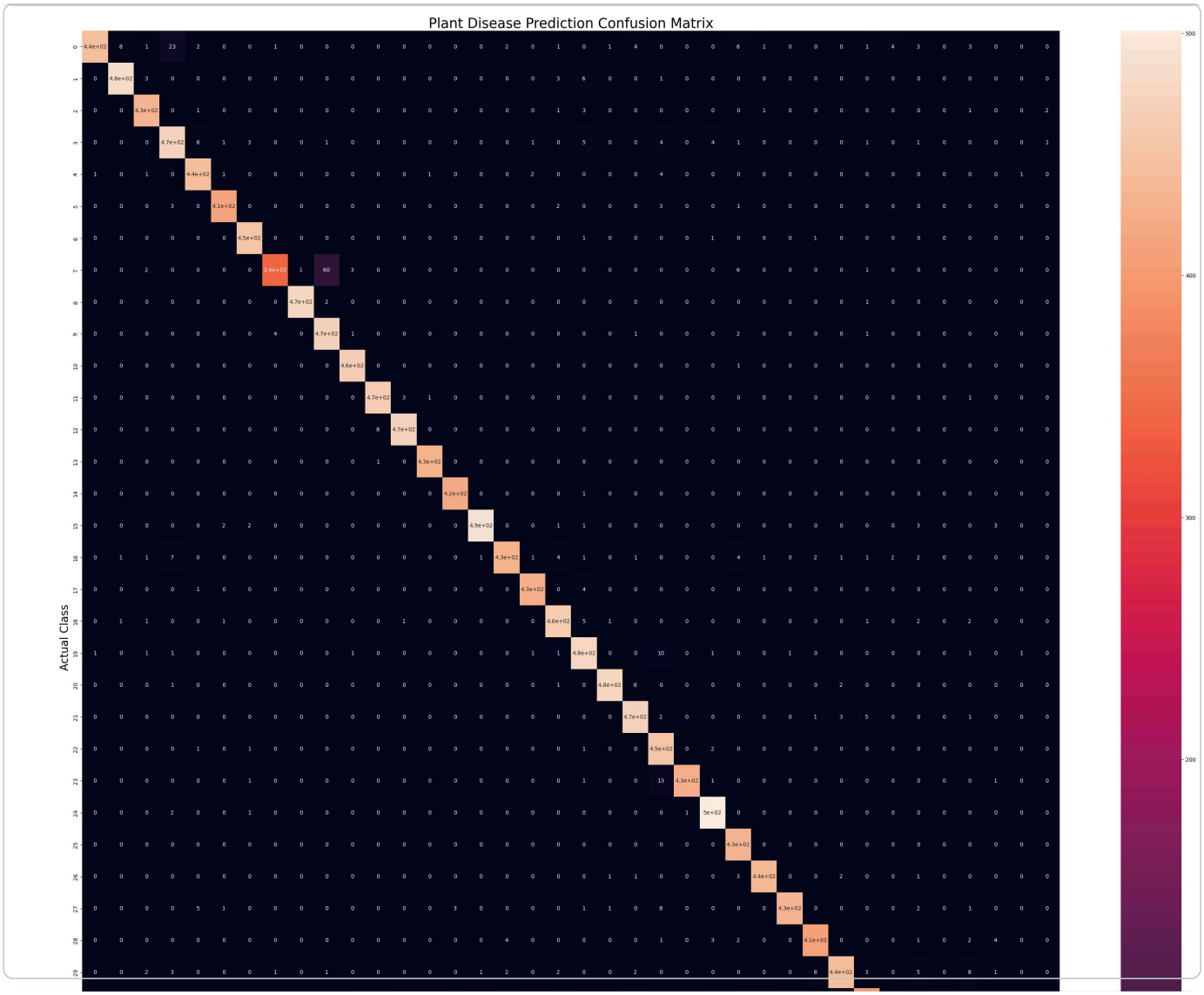
```
                                Pepper,_bell___healthy       0.93      0.96      0.95       497
                                 Potato___Early_blight       0.99      0.98      0.98       485
                                  Potato___Late_blight       0.94      0.98      0.96       485
                                      Potato___healthy       0.89      0.99      0.94       456
                                   Raspberry___healthy       1.00      0.96      0.98       445
                                     Soybean___healthy       0.98      0.99      0.98       505
                               Squash___Powdery_mildew       0.94      1.00      0.97       434
                              Strawberry___Leaf_scorch       0.99      0.98      0.99       444
                                  Strawberry___healthy       1.00      0.95      0.97       456
                                Tomato___Bacterial_spot       0.93      0.96      0.94       425
                                  Tomato___Early_blight       0.92      0.92      0.92       480
                                   Tomato___Late_blight       0.95      0.87      0.91       463
                                     Tomato___Leaf_Mold       0.99      0.89      0.93       470
                            Tomato___Septoria_leaf_spot       0.87      0.90      0.88       436
          Tomato___Spider_mites Two-spotted_spider_mite       0.98      0.86      0.92       435
                                  Tomato___Target_Spot       0.83      0.93      0.88       457
                Tomato___Tomato_Yellow_Leaf_Curl_Virus       0.98      0.98      0.98       490
                          Tomato___Tomato_mosaic_virus       0.99      0.98      0.99       448
                                      Tomato___healthy       0.97      1.00      0.98       481

                                              accuracy                           0.96     17572
                                             macro avg       0.96      0.96      0.96     17572
                                          weighted avg       0.96      0.96      0.96     17572
```

```
cm = confusion_matrix(Y_true,predicted_categories)
cm
```

```
array([[443,   8,   1, ...,   0,   0,   0],
       [  0, 484,   3, ...,   0,   0,   0],
       [  0,   0, 431, ...,   0,   0,   2],
       ...,
       [  0,   0,   2, ..., 481,   0,   0],
       [  0,   0,   0, ...,   0, 441,   0],
       [  0,   0,   0, ...,   0,   0, 480]])
```

```
plt.figure(figsize=(40,40))
sns.heatmap(cm,annot=True,annot_kws={'size':10})
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
500
400
300
200