cornleaf-95-test

March 7, 2024

IMPORTING REQUIRED LIBRARIES

[28]:
```
import pandas as pd
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
import keras
import seaborn as sns
import pathlib
from pathlib import Path
import glob
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils  import to_categorical
from keras.preprocessing.image import ImageDataGenerator
```

[68]:
```
data_dir="/kaggle/input/corn-leaf-disease/corn"
root_dir='/kaggle/working/'
```

[69]:
```
def countfiles(root_dir):
    for path in pathlib.Path(root_dir).iterdir():
        if path.is_dir():
             print( str(len([name for name in os.listdir(path) \
            if os.path.isfile(os.path.join(path, name))])) + " files inside the␣
↪" + \
            str(path.name),'class')
countfiles(data_dir)
```

```
1000 files inside the Corn___Northern_Leaf_Blight class
1162 files inside the Corn___healthy class
1000 files inside the Corn___Cercospora_leaf_spot Gray_leaf_spot class
1192 files inside the Corn___Common_rust class
```

DATASET PREPARATION

[80]:
```
def data_categories(d_path):
    categories=[]     #listdir-->used to get the list of all files and␣
↪directories in the specified directory
```

```
    for folder_name in os.listdir(d_path):  #os.path.isdir()--->used to check␣
↪whether the specified path is an existing directory or not.
        if os.path.isdir(os.path.join(d_path,folder_name)):
            no_of_files = len(glob.glob(os.path.join(d_path, folder_name)+"/*.
↪JPG")) + len(glob.glob(os.path.join(d_path, folder_name)+"/*.jpg"))
            categories.append(np.array([folder_name,no_of_files]))
    categories.sort(key=lambda a:a[0])
    cat=np.array(categories)
    return list(cat[:, 0]),list(cat[:,1])
categories,no_of_files = data_categories("/kaggle/input/corn-leaf-disease/corn")
print(categories)
```

```
['Corn___Cercospora_leaf_spot Gray_leaf_spot', 'Corn___Common_rust',
'Corn___Northern_Leaf_Blight', 'Corn___healthy']
```

[81]:
```
print("number of categories: ", len(categories))
```

```
number of categories:  4
```

[82]:
```
df = pd.DataFrame({"category": categories, "number of files": no_of_files})
df
```

[82]:

	category	number of files
0	Corn___Cercospora_leaf_spot Gray_leaf_spot	1000
1	Corn___Common_rust	1192
2	Corn___Northern_Leaf_Blight	1000
3	Corn___healthy	1162

[83]:
```
def dataset(data_path, categories, width, height):
    x = []
    y = []
    for category_idx, category in enumerate(categories):
        path = os.path.join(data_path, category)
        count = 0
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img))
            img_size = cv2.resize(img_array, (width, height))
            x.append(img_size)
            y.append(category_idx)
            count += 1
        print(f"Number of images in class {category_idx}: {count}")
    y = np.array(y)
    x = np.array(x).reshape(y.shape[0], width, height, 3)
    return x, y
```

```
x, y = dataset(data_path=data_dir, categories=['Corn___Cercospora_leaf_spot␣
↪Gray_leaf_spot', 'Corn___Common_rust', 'Corn___Northern_Leaf_Blight',␣
↪'Corn___healthy'], width=100,height=100)
```

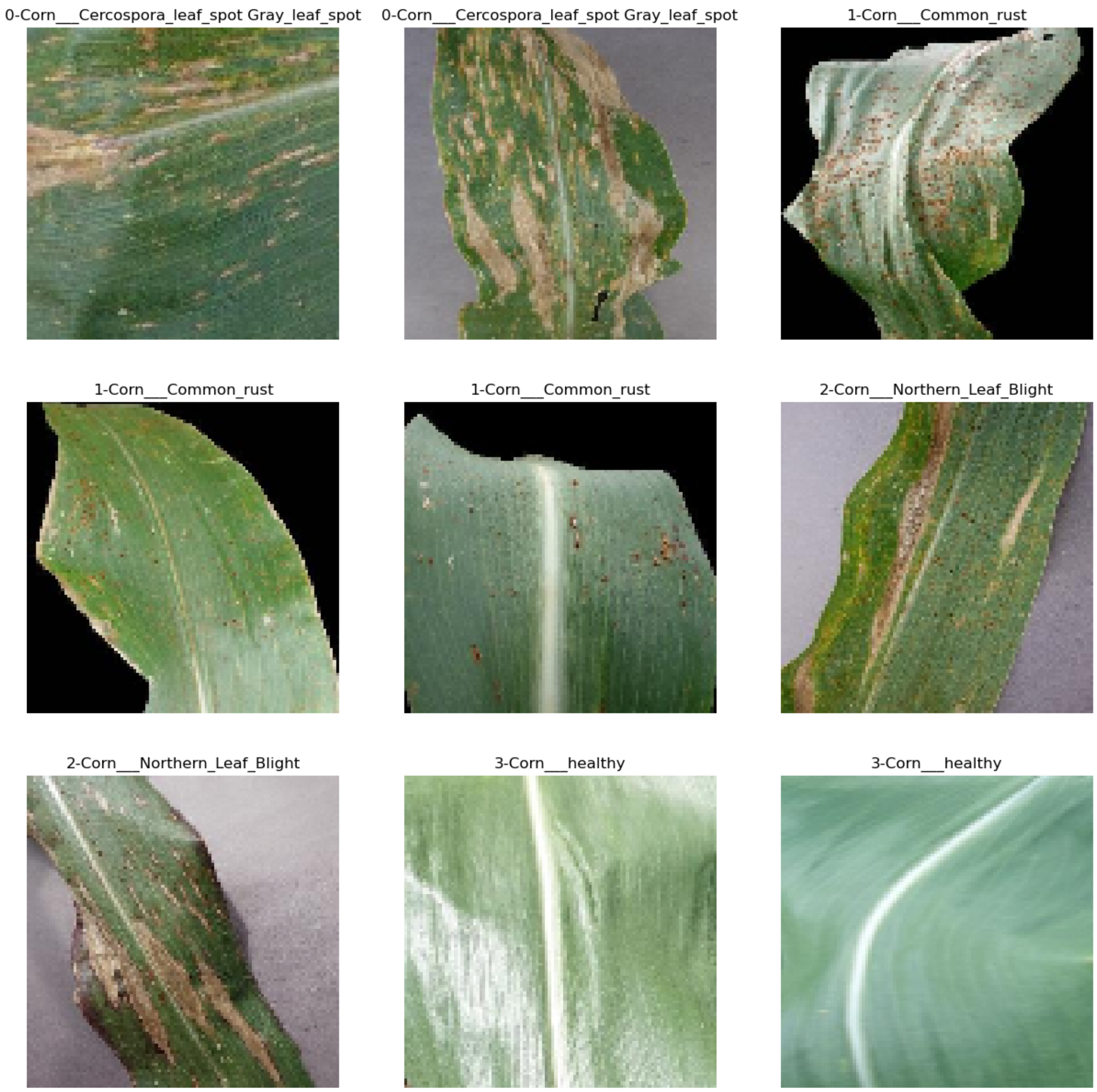
```
Number of images in class 0: 1000
Number of images in class 1: 1192
Number of images in class 2: 1000
Number of images in class 3: 1162
```

[84]:
```
print(f'x shape:{x.shape}')
print(f"y shape: {y.shape}")
```

```
x shape:(4354, 100, 100, 3)
y shape: (4354,)
```

IMAGES FROM CLASSES

[93]:
```
plt.figure(figsize=(15, 15))
st, end = 0,500
for i in range(9):
    plt.subplot(3, 3, i + 1)
    idx = np.random.randint(st, end)
    st = end + 1
    end = (i + 2) * 500
    plt.rcParams.update({'font.size':10})
    plt.imshow(x[idx][:, :, ::-1])
    plt.title(f"{y[idx]}-{categories[y[idx]]}")
    plt.axis("off")
plt.show()
```



DATASET SPLITTING FOR TRAIN/VAL/TEST SETS

[94]:
```
y=np.reshape(y,(len(y),1))
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
    ↪1,random_state=42)
print(f"x_train: {x_train.shape}")
print(f"y_train: {y_train.shape}")
print(f"x_test: {x_test.shape}")
print(f"y_test: {y_test.shape}")
```

```
x_train: (3918, 100, 100, 3)
y_train: (3918, 1)
x_test: (436, 100, 100, 3)
y_test: (436, 1)
```

[95]:
```
x_train,x_val,y_train,y_val=train_test_split(x_train,y_train,train_size=0.70)
x_test=x_test

print(f"x_train:{x_train.shape},y_train:{y_train.shape}")
print(f"x_val: {x_val.shape},y_val:{y_val.shape}")        #70-20-10
print(f"x_test:{x_test.shape},y_test:{y_test.shape}")
```

```
x_train:(2742, 100, 100, 3),y_train:(2742, 1)
x_val: (1176, 100, 100, 3),y_val:(1176, 1)
x_test:(436, 100, 100, 3),y_test:(436, 1)
```

[96]:
```
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)

print(f"x_train:{x_train.shape},  y_train:{y_train.shape}")
print(f"x_val:{x_val.shape},  y_val:{y_val.shape}")
print(f"x_test:{x_test.shape},  y_test:{y_test.shape}")
```

```
x_train:(2742, 100, 100, 3),  y_train:(2742, 4)
x_val:(1176, 100, 100, 3),  y_val:(1176, 4)
x_test:(436, 100, 100, 3),  y_test:(436, 4)
```

DATA PREPROCESSING

[97]:
```
train_generator=ImageDataGenerator(rescale=1./255,
                                    rotation_range=2,
                                   horizontal_flip=True,
                                    shear_range=0.5,
                                   zoom_range=0.7)
val_generator=ImageDataGenerator(rescale=1./255,
                                 rotation_range=2,
                                 horizontal_flip=True,
                                 shear_range=0.5,
                                 zoom_range=0.1)
test_generator=ImageDataGenerator(rotation_range=2,
                                 horizontal_flip=True,
                                 zoom_range=0.1)

train_generator.fit(x_train)
val_generator.fit(x_val)
test_generator.fit(x_test)
```

MODEL BUILDING-CNN

[169]:
```
from keras.models import Sequential, load_model
from keras.layers import Flatten, Dense,MaxPooling2D,Dropout
```

[301]:
```
model = keras.Sequential([
    # Convolutional layers
                   #kernels,  #filters ,  #activation function,          #input
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',␣
 ↪input_shape=(100,100,3)),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Dropout(0.25),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Dropout(0.25),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Dropout(0.25),


    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Dropout(0.25),

    #tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    #tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    #tf.keras.layers.MaxPooling2D((2,2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),#0.2#0.5
    tf.keras.layers.Dense(4, activation='softmax')])
```

[302]:
```
model.summary()
```

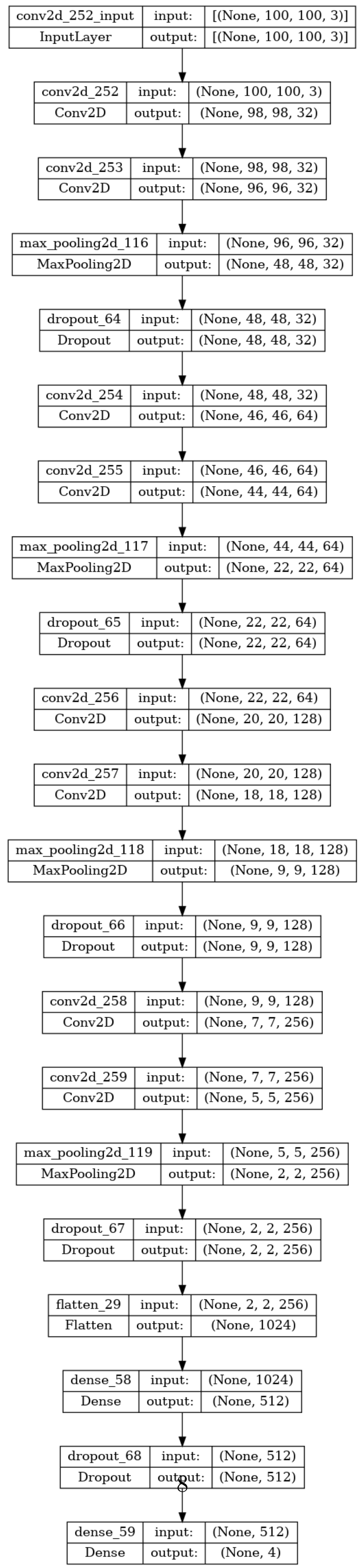
```
Model: "sequential_29"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_252 (Conv2D)         (None, 98, 98, 32)        896

 conv2d_253 (Conv2D)         (None, 96, 96, 32)        9248

 max_pooling2d_116 (MaxPooli  (None, 48, 48, 32)       0
 ng2D)
```

```
 dropout_64 (Dropout)        (None, 48, 48, 32)        0

 conv2d_254 (Conv2D)         (None, 46, 46, 64)        18496

 conv2d_255 (Conv2D)         (None, 44, 44, 64)        36928

 max_pooling2d_117 (MaxPooli  (None, 22, 22, 64)       0
 ng2D)

 dropout_65 (Dropout)        (None, 22, 22, 64)        0

 conv2d_256 (Conv2D)         (None, 20, 20, 128)       73856

 conv2d_257 (Conv2D)         (None, 18, 18, 128)       147584

 max_pooling2d_118 (MaxPooli  (None, 9, 9, 128)        0
 ng2D)

 dropout_66 (Dropout)        (None, 9, 9, 128)         0

 conv2d_258 (Conv2D)         (None, 7, 7, 256)         295168

 conv2d_259 (Conv2D)         (None, 5, 5, 256)         590080

 max_pooling2d_119 (MaxPooli  (None, 2, 2, 256)        0
 ng2D)

 dropout_67 (Dropout)        (None, 2, 2, 256)         0

 flatten_29 (Flatten)        (None, 1024)              0

 dense_58 (Dense)            (None, 512)               524800

 dropout_68 (Dropout)        (None, 512)               0

 dense_59 (Dense)            (None, 4)                 2052

=================================================================
Total params: 1,699,108
Trainable params: 1,699,108
Non-trainable params: 0
_________________________________________________________________
```

[303]:

```python
from tensorflow.keras.utils import plot_model
plot_model(model,show_shapes=True,to_file='4class model.png')
```

[303]:


conv2d_252_input | input: | [(None, 100, 100, 3)]
InputLayer | output: | [(None, 100, 100, 3)]
conv2d_252 | input: | (None, 100, 100, 3)
Conv2D | output: | (None, 98, 98, 32)
conv2d_253 | input: | (None, 98, 98, 32)
Conv2D | output: | (None, 96, 96, 32)
max_pooling2d_116 | input: | (None, 96, 96, 32)
MaxPooling2D | output: | (None, 48, 48, 32)
dropout_64 | input: | (None, 48, 48, 32)
Dropout | output: | (None, 48, 48, 32)
conv2d_254 | input: | (None, 48, 48, 32)
Conv2D | output: | (None, 46, 46, 64)
conv2d_255 | input: | (None, 46, 46, 64)
Conv2D | output: | (None, 44, 44, 64)
max_pooling2d_117 | input: | (None, 44, 44, 64)
MaxPooling2D | output: | (None, 22, 22, 64)
dropout_65 | input: | (None, 22, 22, 64)
Dropout | output: | (None, 22, 22, 64)
conv2d_256 | input: | (None, 22, 22, 64)
Conv2D | output: | (None, 20, 20, 128)
conv2d_257 | input: | (None, 20, 20, 128)
Conv2D | output: | (None, 18, 18, 128)
max_pooling2d_118 | input: | (None, 18, 18, 128)
MaxPooling2D | output: | (None, 9, 9, 128)
dropout_66 | input: | (None, 9, 9, 128)
Dropout | output: | (None, 9, 9, 128)
conv2d_258 | input: | (None, 9, 9, 128)
Conv2D | output: | (None, 7, 7, 256)
conv2d_259 | input: | (None, 7, 7, 256)
Conv2D | output: | (None, 5, 5, 256)
max_pooling2d_119 | input: | (None, 5, 5, 256)
MaxPooling2D | output: | (None, 2, 2, 256)
dropout_67 | input: | (None, 2, 2, 256)
Dropout | output: | (None, 2, 2, 256)
flatten_29 | input: | (None, 2, 2, 256)
Flatten | output: | (None, 1024)
dense_58 | input: | (None, 1024)
Dense | output: | (None, 512)
dropout_68 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_59 | input: | (None, 512)
Dense | output: | (None, 4)

[304]:
```
from keras.metrics import Precision,Recall
import tensorflow_addons as tfa
```

[305]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
             ␣
 ↪metrics=['accuracy',Precision(name='precision'),Recall(name='Recall'),tfa.
 ↪metrics.F1Score(num_classes=4)])
```

[306]:
```
from tensorflow.keras.callbacks import ModelCheckpoint

model_filepath='/kaggle/working/4m-{epoch:02d}--{accuracy:4f}.hdf5'
checkpt=ModelCheckpoint(filepath=model_filepath,
                        monitor='accuracy',
                        mode='max',
                        save_best_only=True)
```

[307]:
```
history = model.fit(x_train,y_train, epochs=100,batch_size=120,
                    validation_data = val_generator.
 ↪flow(x_val,y_val,batch_size=120),
                   validation_steps=200,
                   verbose=1,callbacks=[checkpt])
history=history.history
model.save('/kaggle/working/4c_model.h5')
```

```
Epoch 1/100

2023-02-26 06:02:24.462056: E
tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed:
INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin
shape insequential_29/dropout_64/dropout/SelectV2-2-TransposeNHWCToNCHW-
LayoutOptimizer

23/23 [==============================] - 9s 236ms/step - loss: 4.3169 -
accuracy: 0.2695 - precision: 0.2834 - Recall: 0.0565 - f1_score: 0.2216 -
val_loss: 1.3864 - val_accuracy: 0.2372 - val_precision: 0.0000e+00 -
val_Recall: 0.0000e+00 - val_f1_score: 0.0959
Epoch 2/100
23/23 [==============================] - 1s 62ms/step - loss: 1.2175 - accuracy:
0.4096 - precision: 0.6737 - Recall: 0.1627 - f1_score: 0.3402
Epoch 3/100
23/23 [==============================] - 1s 64ms/step - loss: 1.0078 - accuracy:
0.5219 - precision: 0.7139 - Recall: 0.2994 - f1_score: 0.4949
Epoch 4/100
23/23 [==============================] - 1s 61ms/step - loss: 0.7425 - accuracy:
0.6816 - precision: 0.8013 - Recall: 0.5573 - f1_score: 0.6553
```

```
Epoch 5/100
23/23 [==============================] - 1s 62ms/step - loss: 0.5582 - accuracy:
0.7699 - precision: 0.8551 - Recall: 0.6802 - f1_score: 0.7504
Epoch 6/100
23/23 [==============================] - 1s 61ms/step - loss: 0.4282 - accuracy:
0.8344 - precision: 0.8752 - Recall: 0.7673 - f1_score: 0.8214
Epoch 7/100
23/23 [==============================] - 1s 61ms/step - loss: 0.3620 - accuracy:
0.8578 - precision: 0.8782 - Recall: 0.8253 - f1_score: 0.8464
Epoch 8/100
23/23 [==============================] - 1s 62ms/step - loss: 0.3021 - accuracy:
0.8716 - precision: 0.8824 - Recall: 0.8589 - f1_score: 0.8612
Epoch 9/100
23/23 [==============================] - 1s 61ms/step - loss: 0.2991 - accuracy:
0.8804 - precision: 0.8881 - Recall: 0.8687 - f1_score: 0.8710
Epoch 10/100
23/23 [==============================] - 1s 62ms/step - loss: 0.2670 - accuracy:
0.9001 - precision: 0.9089 - Recall: 0.8917 - f1_score: 0.8918
Epoch 11/100
23/23 [==============================] - 1s 64ms/step - loss: 0.2423 - accuracy:
0.9048 - precision: 0.9081 - Recall: 0.9008 - f1_score: 0.8975
Epoch 12/100
23/23 [==============================] - 1s 61ms/step - loss: 0.2094 - accuracy:
0.9176 - precision: 0.9203 - Recall: 0.9143 - f1_score: 0.9107
Epoch 13/100
23/23 [==============================] - 1s 57ms/step - loss: 0.2181 - accuracy:
0.9172 - precision: 0.9201 - Recall: 0.9150 - f1_score: 0.9102
Epoch 14/100
23/23 [==============================] - 1s 57ms/step - loss: 0.2214 - accuracy:
0.9143 - precision: 0.9178 - Recall: 0.9121 - f1_score: 0.9076
Epoch 15/100
23/23 [==============================] - 1s 58ms/step - loss: 0.2365 - accuracy:
0.9052 - precision: 0.9079 - Recall: 0.9019 - f1_score: 0.8981
Epoch 16/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1917 - accuracy:
0.9212 - precision: 0.9243 - Recall: 0.9179 - f1_score: 0.9151
Epoch 17/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1783 - accuracy:
0.9300 - precision: 0.9312 - Recall: 0.9282 - f1_score: 0.9242
Epoch 18/100
23/23 [==============================] - 1s 58ms/step - loss: 0.2458 - accuracy:
0.9055 - precision: 0.9096 - Recall: 0.8990 - f1_score: 0.8983
Epoch 19/100
23/23 [==============================] - 1s 60ms/step - loss: 0.1887 - accuracy:
0.9256 - precision: 0.9274 - Recall: 0.9227 - f1_score: 0.9194
Epoch 20/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1724 - accuracy:
0.9303 - precision: 0.9316 - Recall: 0.9292 - f1_score: 0.9246
```

```
Epoch 21/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1638 - accuracy:
0.9354 - precision: 0.9367 - Recall: 0.9344 - f1_score: 0.9301
Epoch 22/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1389 - accuracy:
0.9457 - precision: 0.9459 - Recall: 0.9438 - f1_score: 0.9409
Epoch 23/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1440 - accuracy:
0.9435 - precision: 0.9447 - Recall: 0.9413 - f1_score: 0.9388
Epoch 24/100
23/23 [==============================] - 1s 63ms/step - loss: 0.1303 - accuracy:
0.9508 - precision: 0.9518 - Recall: 0.9504 - f1_score: 0.9467
Epoch 25/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1643 - accuracy:
0.9362 - precision: 0.9392 - Recall: 0.9347 - f1_score: 0.9311
Epoch 26/100
23/23 [==============================] - 1s 57ms/step - loss: 0.2064 - accuracy:
0.9296 - precision: 0.9333 - Recall: 0.9238 - f1_score: 0.9246
Epoch 27/100
23/23 [==============================] - 1s 59ms/step - loss: 0.1790 - accuracy:
0.9234 - precision: 0.9254 - Recall: 0.9227 - f1_score: 0.9169
Epoch 28/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1589 - accuracy:
0.9409 - precision: 0.9421 - Recall: 0.9384 - f1_score: 0.9362
Epoch 29/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1514 - accuracy:
0.9457 - precision: 0.9463 - Recall: 0.9449 - f1_score: 0.9413
Epoch 30/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1359 - accuracy:
0.9478 - precision: 0.9482 - Recall: 0.9471 - f1_score: 0.9437
Epoch 31/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1638 - accuracy:
0.9358 - precision: 0.9364 - Recall: 0.9340 - f1_score: 0.9306
Epoch 32/100
23/23 [==============================] - 1s 58ms/step - loss: 0.1564 - accuracy:
0.9435 - precision: 0.9454 - Recall: 0.9409 - f1_score: 0.9392
Epoch 33/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1350 - accuracy:
0.9446 - precision: 0.9459 - Recall: 0.9442 - f1_score: 0.9400
Epoch 34/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1000 - accuracy:
0.9635 - precision: 0.9642 - Recall: 0.9635 - f1_score: 0.9605
Epoch 35/100
23/23 [==============================] - 1s 64ms/step - loss: 0.0882 - accuracy:
0.9668 - precision: 0.9671 - Recall: 0.9661 - f1_score: 0.9640
Epoch 36/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1094 - accuracy:
0.9610 - precision: 0.9613 - Recall: 0.9602 - f1_score: 0.9577
```

```
Epoch 37/100
23/23 [==============================] - 1s 58ms/step - loss: 0.1418 - accuracy:
0.9537 - precision: 0.9550 - Recall: 0.9526 - f1_score: 0.9499
Epoch 38/100
23/23 [==============================] - 1s 57ms/step - loss: 0.2166 - accuracy:
0.9409 - precision: 0.9427 - Recall: 0.9354 - f1_score: 0.9374
Epoch 39/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1387 - accuracy:
0.9478 - precision: 0.9526 - Recall: 0.9449 - f1_score: 0.9443
Epoch 40/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1065 - accuracy:
0.9562 - precision: 0.9572 - Recall: 0.9551 - f1_score: 0.9528
Epoch 41/100
23/23 [==============================] - 1s 58ms/step - loss: 0.1144 - accuracy:
0.9599 - precision: 0.9601 - Recall: 0.9570 - f1_score: 0.9566
Epoch 42/100
23/23 [==============================] - 1s 61ms/step - loss: 0.0876 - accuracy:
0.9679 - precision: 0.9679 - Recall: 0.9668 - f1_score: 0.9654
Epoch 43/100
23/23 [==============================] - 1s 63ms/step - loss: 0.0940 - accuracy:
0.9697 - precision: 0.9704 - Recall: 0.9694 - f1_score: 0.9672
Epoch 44/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1166 - accuracy:
0.9540 - precision: 0.9558 - Recall: 0.9537 - f1_score: 0.9502
Epoch 45/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1343 - accuracy:
0.9508 - precision: 0.9528 - Recall: 0.9497 - f1_score: 0.9468
Epoch 46/100
23/23 [==============================] - 1s 59ms/step - loss: 0.1676 - accuracy:
0.9431 - precision: 0.9465 - Recall: 0.9413 - f1_score: 0.9390
Epoch 47/100
23/23 [==============================] - 1s 61ms/step - loss: 0.1013 - accuracy:
0.9661 - precision: 0.9675 - Recall: 0.9657 - f1_score: 0.9636
Epoch 48/100
23/23 [==============================] - 1s 57ms/step - loss: 0.3725 - accuracy:
0.9577 - precision: 0.9601 - Recall: 0.9555 - f1_score: 0.9551
Epoch 49/100
23/23 [==============================] - 1s 57ms/step - loss: 0.1988 - accuracy:
0.9384 - precision: 0.9425 - Recall: 0.9322 - f1_score: 0.9347
Epoch 50/100
23/23 [==============================] - 1s 61ms/step - loss: 0.0849 - accuracy:
0.9705 - precision: 0.9726 - Recall: 0.9694 - f1_score: 0.9681
Epoch 51/100
23/23 [==============================] - 1s 64ms/step - loss: 0.0774 - accuracy:
0.9730 - precision: 0.9730 - Recall: 0.9730 - f1_score: 0.9708
Epoch 52/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0825 - accuracy:
0.9708 - precision: 0.9726 - Recall: 0.9701 - f1_score: 0.9686
```

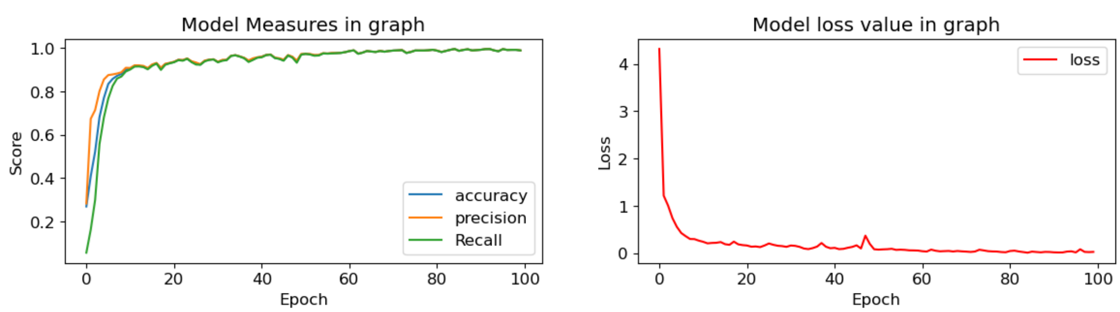
```
Epoch 53/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0861 - accuracy:
0.9664 - precision: 0.9678 - Recall: 0.9643 - f1_score: 0.9643
Epoch 54/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0947 - accuracy:
0.9664 - precision: 0.9682 - Recall: 0.9654 - f1_score: 0.9639
Epoch 55/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0720 - accuracy:
0.9752 - precision: 0.9763 - Recall: 0.9748 - f1_score: 0.9733
Epoch 56/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0766 - accuracy:
0.9745 - precision: 0.9748 - Recall: 0.9737 - f1_score: 0.9724
Epoch 57/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0719 - accuracy:
0.9759 - precision: 0.9773 - Recall: 0.9748 - f1_score: 0.9739
Epoch 58/100
23/23 [==============================] - 1s 61ms/step - loss: 0.0619 - accuracy:
0.9767 - precision: 0.9770 - Recall: 0.9759 - f1_score: 0.9748
Epoch 59/100
23/23 [==============================] - 1s 63ms/step - loss: 0.0604 - accuracy:
0.9774 - precision: 0.9777 - Recall: 0.9774 - f1_score: 0.9755
Epoch 60/100
23/23 [==============================] - 1s 61ms/step - loss: 0.0573 - accuracy:
0.9807 - precision: 0.9807 - Recall: 0.9807 - f1_score: 0.9791
Epoch 61/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0434 - accuracy:
0.9858 - precision: 0.9861 - Recall: 0.9858 - f1_score: 0.9846
Epoch 62/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0367 - accuracy:
0.9891 - precision: 0.9894 - Recall: 0.9891 - f1_score: 0.9881
Epoch 63/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0770 - accuracy:
0.9737 - precision: 0.9741 - Recall: 0.9730 - f1_score: 0.9718
Epoch 64/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0541 - accuracy:
0.9788 - precision: 0.9788 - Recall: 0.9785 - f1_score: 0.9773
Epoch 65/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0413 - accuracy:
0.9861 - precision: 0.9865 - Recall: 0.9861 - f1_score: 0.9851
Epoch 66/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0454 - accuracy:
0.9836 - precision: 0.9839 - Recall: 0.9836 - f1_score: 0.9822
Epoch 67/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0490 - accuracy:
0.9818 - precision: 0.9818 - Recall: 0.9818 - f1_score: 0.9803
Epoch 68/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0381 - accuracy:
0.9869 - precision: 0.9872 - Recall: 0.9861 - f1_score: 0.9860
```

```
Epoch 69/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0483 - accuracy:
0.9829 - precision: 0.9832 - Recall: 0.9829 - f1_score: 0.9817
Epoch 70/100
23/23 [==============================] - 1s 61ms/step - loss: 0.0403 - accuracy:
0.9861 - precision: 0.9861 - Recall: 0.9861 - f1_score: 0.9851
Epoch 71/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0356 - accuracy:
0.9883 - precision: 0.9883 - Recall: 0.9880 - f1_score: 0.9874
Epoch 72/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0283 - accuracy:
0.9891 - precision: 0.9891 - Recall: 0.9891 - f1_score: 0.9881
Epoch 73/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0372 - accuracy:
0.9905 - precision: 0.9909 - Recall: 0.9898 - f1_score: 0.9898
Epoch 74/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0749 - accuracy:
0.9770 - precision: 0.9777 - Recall: 0.9770 - f1_score: 0.9758
Epoch 75/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0603 - accuracy:
0.9821 - precision: 0.9821 - Recall: 0.9814 - f1_score: 0.9807
Epoch 76/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0444 - accuracy:
0.9887 - precision: 0.9887 - Recall: 0.9883 - f1_score: 0.9879
Epoch 77/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0388 - accuracy:
0.9883 - precision: 0.9887 - Recall: 0.9883 - f1_score: 0.9875
Epoch 78/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0369 - accuracy:
0.9883 - precision: 0.9887 - Recall: 0.9883 - f1_score: 0.9874
Epoch 79/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0260 - accuracy:
0.9894 - precision: 0.9894 - Recall: 0.9894 - f1_score: 0.9886
Epoch 80/100
23/23 [==============================] - 1s 60ms/step - loss: 0.0214 - accuracy:
0.9920 - precision: 0.9920 - Recall: 0.9920 - f1_score: 0.9913
Epoch 81/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0479 - accuracy:
0.9876 - precision: 0.9883 - Recall: 0.9876 - f1_score: 0.9867
Epoch 82/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0555 - accuracy:
0.9814 - precision: 0.9821 - Recall: 0.9814 - f1_score: 0.9799
Epoch 83/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0380 - accuracy:
0.9865 - precision: 0.9869 - Recall: 0.9865 - f1_score: 0.9855
Epoch 84/100
23/23 [==============================] - 1s 59ms/step - loss: 0.0255 - accuracy:
0.9916 - precision: 0.9916 - Recall: 0.9916 - f1_score: 0.9909
```

```
Epoch 85/100
23/23 [==============================] - 1s 62ms/step - loss: 0.0124 - accuracy:
0.9956 - precision: 0.9956 - Recall: 0.9956 - f1_score: 0.9953
Epoch 86/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0354 - accuracy:
0.9869 - precision: 0.9869 - Recall: 0.9869 - f1_score: 0.9860
Epoch 87/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0271 - accuracy:
0.9912 - precision: 0.9920 - Recall: 0.9912 - f1_score: 0.9905
Epoch 88/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0197 - accuracy:
0.9942 - precision: 0.9942 - Recall: 0.9942 - f1_score: 0.9937
Epoch 89/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0294 - accuracy:
0.9887 - precision: 0.9887 - Recall: 0.9887 - f1_score: 0.9878
Epoch 90/100
23/23 [==============================] - 1s 56ms/step - loss: 0.0275 - accuracy:
0.9905 - precision: 0.9905 - Recall: 0.9902 - f1_score: 0.9897
Epoch 91/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0199 - accuracy:
0.9920 - precision: 0.9920 - Recall: 0.9920 - f1_score: 0.9913
Epoch 92/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0172 - accuracy:
0.9949 - precision: 0.9953 - Recall: 0.9949 - f1_score: 0.9945
Epoch 93/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0179 - accuracy:
0.9953 - precision: 0.9953 - Recall: 0.9953 - f1_score: 0.9949
Epoch 94/100
23/23 [==============================] - 1s 59ms/step - loss: 0.0365 - accuracy:
0.9891 - precision: 0.9891 - Recall: 0.9887 - f1_score: 0.9883
Epoch 95/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0411 - accuracy:
0.9850 - precision: 0.9858 - Recall: 0.9850 - f1_score: 0.9841
Epoch 96/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0185 - accuracy:
0.9949 - precision: 0.9949 - Recall: 0.9949 - f1_score: 0.9945
Epoch 97/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0848 - accuracy:
0.9912 - precision: 0.9912 - Recall: 0.9909 - f1_score: 0.9907
Epoch 98/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0295 - accuracy:
0.9916 - precision: 0.9916 - Recall: 0.9916 - f1_score: 0.9910
Epoch 99/100
23/23 [==============================] - 1s 57ms/step - loss: 0.0251 - accuracy:
0.9916 - precision: 0.9920 - Recall: 0.9916 - f1_score: 0.9909
Epoch 100/100
23/23 [==============================] - 1s 58ms/step - loss: 0.0295 - accuracy:
0.9887 - precision: 0.9891 - Recall: 0.9887 - f1_score: 0.9879
```

CNN MODEL EVALUATION

[308]:
```
fig = plt.figure(figsize=(14, 3))
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(history['accuracy'])
ax1.plot(history['precision'])
ax1.plot(history["Recall"])
ax1.legend(['accuracy','precision','Recall'])
ax1.set_title('Model Measures in graph')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Score')

ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(history['loss'],color='red')
ax2.legend(['loss'])
ax2.set_title('Model loss value in graph')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')

plt.savefig('graph.png')
plt.show()
```



TEST SET

[332]:
```
best=load_model("/kaggle/working/4m-85--0.995624.hdf5")
testscore=best.evaluate(x_test,y_test)
testscore
```

```
14/14 [==============================] - 0s 7ms/step - loss: 0.2278 - accuracy:
0.9518 - precision: 0.9517 - Recall: 0.9495 - f1_score: 0.9461
```

[332]:
```
[0.2278108149766922,
 0.9518348574638367,
 0.951724112033844,
 0.9495412707328796,
 array([0.90666664, 1.        , 0.88622755, 0.991453  ], dtype=float32)]
```

[333]:
```
print('TEST DATA')
print('')
print(f"Accuracy: {round(testscore[1]*100,2)}%")
print(f"Precision: {round(testscore[2]*100,2)}%")
print(f"Recall: {round(testscore[3]*100,2)}%")
print(f"F1_score: {testscore[4]}")
print(f"Loss: {testscore[0]}")
```

```
TEST DATA

Accuracy: 95.18%
Precision: 95.17%
Recall: 94.95%
F1_score: [0.90666664 1.         0.88622755 0.991453  ]
Loss: 0.2278108149766922
```

[334]:
```
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

[328]:
```
class_names=['Corn___Cercospora_leaf_spot Gray_leaf_spot',␣
 ↪'Corn___Common_rust', 'Corn___Northern_Leaf_Blight', 'Corn___healthy']
```

[335]:
```
y_pred=np.argmax(best.predict(x_test),axis=1)
y_true=np.argmax(y_test,axis=1)
```

```
14/14 [==============================] - 0s 4ms/step
```

[336]:
```
c_test=confusion_matrix(y_true,y_pred)
c_test
```

[336]:
```
array([[102,   0,   7,   0],
       [  0, 123,   0,   0],
       [ 12,   0,  74,   0],
       [  2,   0,   0, 116]])
```

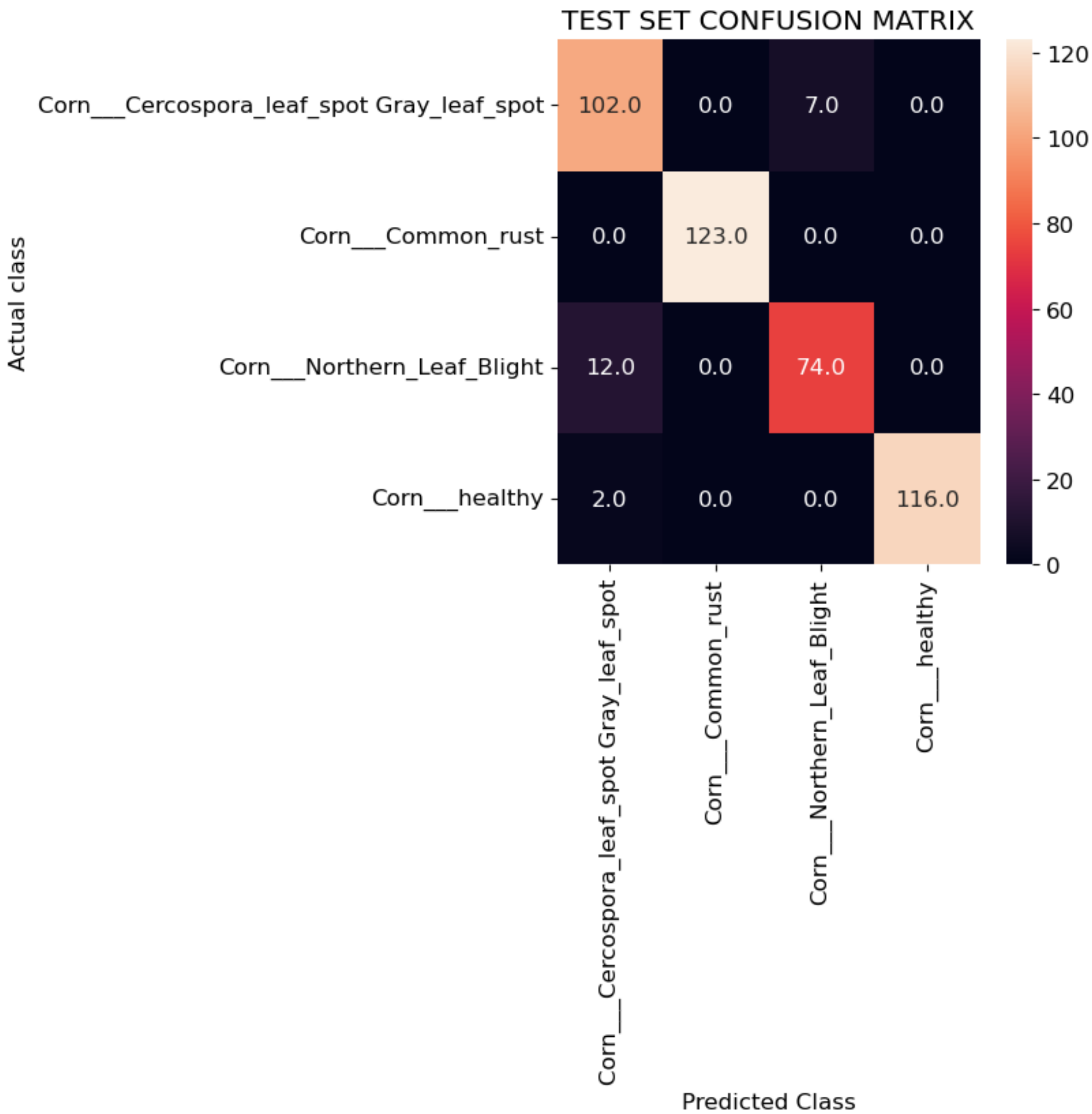
[337]:
```
print(classification_report(y_true,y_pred,target_names=class_names))
```

```
                                            precision    recall  f1-score
support

Corn___Cercospora_leaf_spot Gray_leaf_spot       0.88      0.94      0.91
109
                        Corn___Common_rust       1.00      1.00      1.00
123
               Corn___Northern_Leaf_Blight       0.91      0.86      0.89
86
                            Corn___healthy       1.00      0.98      0.99
118
```

```
                                   accuracy                            0.95
436
                                  macro avg       0.95       0.94      0.95
436
                               weighted avg       0.95       0.95      0.95
436
```

[338]:

```
plt.figure(figsize=(5,5))
sns.
 ↪heatmap(c_test,annot=True,xticklabels=class_names,yticklabels=class_names,fmt='.
 ↪1f')
plt.title('TEST SET CONFUSION MATRIX')
plt.xlabel("Predicted Class")
plt.ylabel("Actual class")
plt.savefig('cm test.png')
plt.show()
```



[348]:
```
print(f"TEST SET")
print('')
for i in range(4):
    tp = c_test[i, i]
    tn = np.sum(c_test) - np.sum(c_test[i, :]) - np.sum(c_test[:, i]) +
 ↪c_test[i, i]
    fp = np.sum(c_test[:, i]) - c_test[i, i]
    fn = np.sum(c_test[i, :]) - c_test[i, i]
    print(f"Class {i}: TP={tp}, TN={tn}, FP={fp}, FN={fn}")
```

```
TEST SET
```

```
Class 0: TP=102, TN=313, FP=14, FN=7
Class 1: TP=123, TN=313, FP=0, FN=0
Class 2: TP=74, TN=343, FP=7, FN=12
Class 3: TP=116, TN=318, FP=0, FN=2
```

VALIDATION SET ANALYSIS

[340]:
```
valscore=best.evaluate(x_val,y_val)
valscore
```

```
37/37 [==============================] - 0s 7ms/step - loss: 8.6482 - accuracy:
0.9422 - precision: 0.9430 - Recall: 0.9422 - f1_score: 0.9381
```

[340]:
```
[8.64823055267334,
 0.942176878452301,
 0.9429787397384644,
 0.942176878452301,
 array([0.88     , 0.9969419, 0.8802946, 0.9950413], dtype=float32)]
```

[341]:
```
print('VALIDATION DATA')
print('')
print(f"Accuracy: {round(valscore[1]*100,2)}%")
print(f"Precision: {round(valscore[2]*100,2)}%")
print(f"Recall: {round(valscore[3]*100,2)}%")
print(f"F1_score: {valscore[4]}")
print(f"Loss: {valscore[0]}")
```

```
VALIDATION DATA

Accuracy: 94.22%
Precision: 94.3%
Recall: 94.22%
F1_score: [0.88      0.9969419 0.8802946 0.9950413]
Loss: 8.64823055267334
```

[342]:
```
yv_pred=np.argmax(best.predict(x_val),axis=1)
yv_true=np.argmax(y_val,axis=1)
```

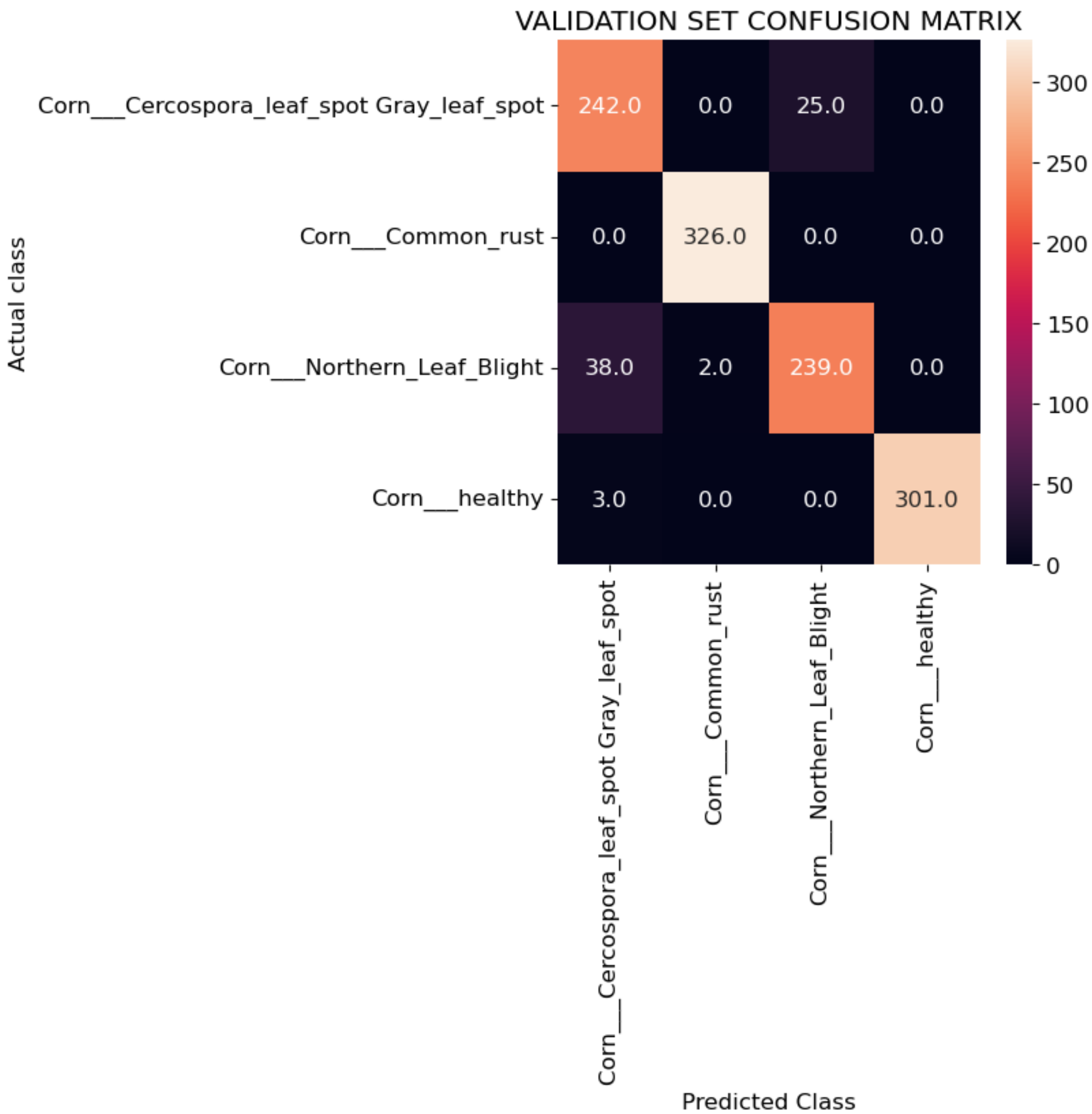
```
37/37 [==============================] - 0s 4ms/step
```

[349]:
```
c_val=confusion_matrix(yv_true,yv_pred)
c_val
```

[349]:
```
array([[242,   0,  25,   0],
       [  0, 326,   0,   0],
       [ 38,   2, 239,   0],
       [  3,   0,   0, 301]])
```

[350]:

```
plt.figure(figsize=(5,5))
sns.
 ↪heatmap(c_val,annot=True,xticklabels=class_names,yticklabels=class_names,fmt='.
 ↪1f')
plt.title('VALIDATION SET CONFUSION MATRIX')
plt.xlabel("Predicted Class")
plt.ylabel("Actual class")
plt.savefig('cm VAL.png')
plt.show()
```



[351]:

```
print(classification_report(yv_true,yv_pred,target_names=class_names))
```

```
                                            precision    recall  f1-score
support

Corn___Cercospora_leaf_spot Gray_leaf_spot       0.86      0.91      0.88
267
                        Corn___Common_rust       0.99      1.00      1.00
326
               Corn___Northern_Leaf_Blight       0.91      0.86      0.88
279
                            Corn___healthy       1.00      0.99      1.00
304

                                  accuracy                           0.94
1176
                                 macro avg       0.94      0.94      0.94
1176
                              weighted avg       0.94      0.94      0.94
1176
```

[352]:
```
print(f"VALIDATION SET")
print('')
for i in range(4):
    tp = c_val[i, i]
    tn = np.sum(c_val) - np.sum(c_val[i, :]) - np.sum(c_val[:, i]) + c_val[i, i]
    fp = np.sum(c_val[:, i]) - c_val[i, i]
    fn = np.sum(c_val[i, :]) - c_val[i, i]
    print(f"Class {i}: TP={tp}, TN={tn}, FP={fp}, FN={fn}")
```

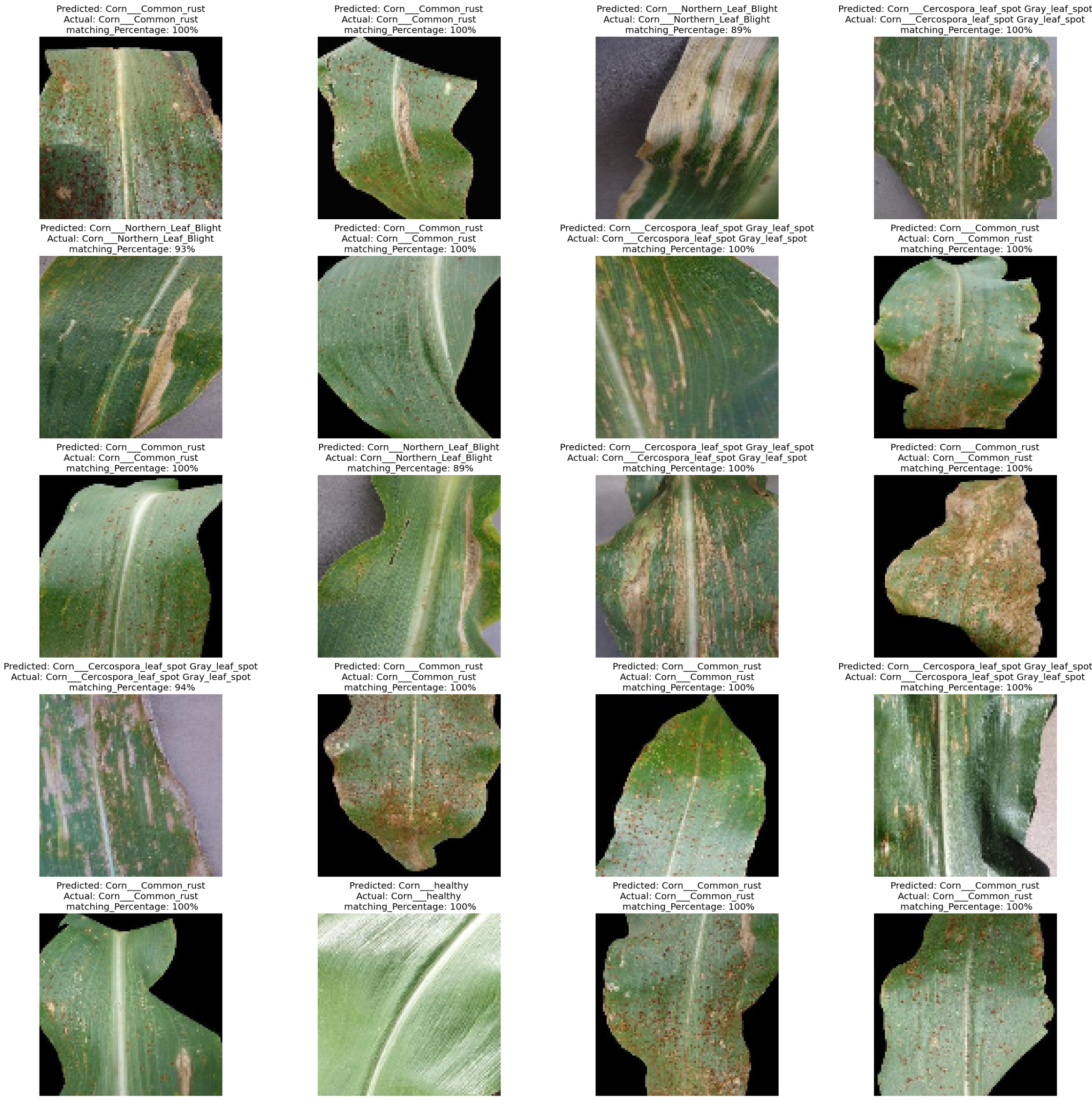
```
VALIDATION SET

Class 0: TP=242, TN=868, FP=41, FN=25
Class 1: TP=326, TN=848, FP=2, FN=0
Class 2: TP=239, TN=872, FP=25, FN=40
Class 3: TP=301, TN=872, FP=0, FN=3
```

IMAGE PREDICTIONS WITH PERCENTAGES

[353]:
```
plt.figure(figsize=(30, 30))
plt.subplots_adjust(wspace=0.2, hspace=0.2)
for i in range(20):
    idx = np.random.randint(len(y))
    img, true_class = x[idx], categories[y[idx].squeeze()]

    # predict class probabilities for the current image
    probs = model.predict(img[None, :, :, :])[0]
    pred_class = categories[np.argmax(probs)]
    max_prob = np.max(probs)*100
```

```
    plt.rcParams.update({'font.size':12})
    plt.subplot(5, 4, i + 1)
    plt.imshow(img[:, :, ::-1])
    plt.title(f"Predicted: {pred_class}\nActual: {true_class}\n␣
↪matching_Percentage: {round(max_prob)}%")
    plt.axis("off")
plt.show()
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
```


Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Northern_Leaf_Blight
Actual: Corn___Northern_Leaf_Blight
matching_Percentage: 89%
Predicted: Corn___Cercospora_leaf_spot Gray_leaf_spot
Actual: Corn___Cercospora_leaf_spot Gray_leaf_spot
matching_Percentage: 100%
Predicted: Corn___Northern_Leaf_Blight
Actual: Corn___Northern_Leaf_Blight
matching_Percentage: 93%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Cercospora_leaf_spot Gray_leaf_spot
Actual: Corn___Cercospora_leaf_spot Gray_leaf_spot
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Northern_Leaf_Blight
Actual: Corn___Northern_Leaf_Blight
matching_Percentage: 89%
Predicted: Corn___Cercospora_leaf_spot Gray_leaf_spot
Actual: Corn___Cercospora_leaf_spot Gray_leaf_spot
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Cercospora_leaf_spot Gray_leaf_spot
Actual: Corn___Cercospora_leaf_spot Gray_leaf_spot
matching_Percentage: 94%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Cercospora_leaf_spot Gray_leaf_spot
Actual: Corn___Cercospora_leaf_spot Gray_leaf_spot
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___healthy
Actual: Corn___healthy
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%
Predicted: Corn___Common_rust
Actual: Corn___Common_rust
matching_Percentage: 100%