# APPENDIX-II (SOURCE CODE)

**#Import Libraries** import plaidml.keras import os

plaidml.keras.install\_backend() os.environ["KERAS\_BACKEND"] = "plaidml.keras.backend"

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img from tensorflow.keras.preprocessing import image

from tensorflow.keras.optimizers import RMSprop from keras.utils import to\_categorical

from sklearn.model\_selection import train\_test\_split from matplotlib.pyplot import imshow

from PIL import Image

from IPython.display import Image import matplotlib.pyplot as plt import tensorflow as tf

import numpy as np import pandas as pd import random import cv2

import os

#### #Import Training Dataset

filenames = os.listdir('E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data')

categories = []

for filename in filenames: category = filename.split('.')[0] if category == 'pla':

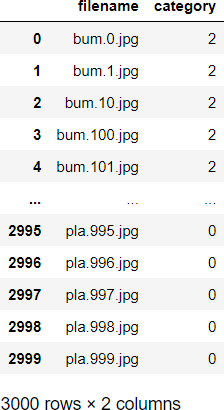
categories.append(0) elif category == 'den':

categories.append(1) elif category == 'bum':

categories.append(2) df = pd.DataFrame({

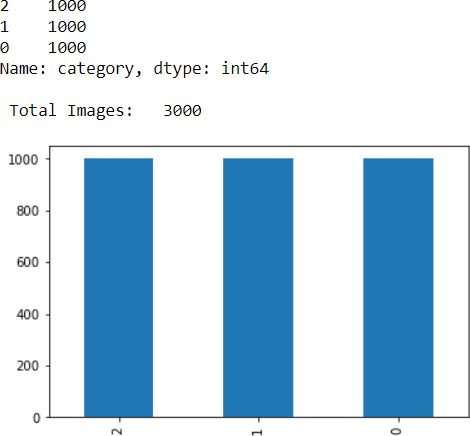
'filename' : filenames, 'category' : categories})

df



**#Plotting All Training Data (Category Wise)** df['category'].value\_counts().plot.bar() print(df['category'].value\_counts())

print("\n Total Images: ", df.shape[0])



#### #Proposed CNN Model

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Activation, Dense, Flatten,

Dropout

model2 = Sequential()

model2.add(Conv2D(32, (3, 3), activation ='relu', input\_shape = (150,150,3))) model2.add(MaxPooling2D(pool\_size=(2,2)))

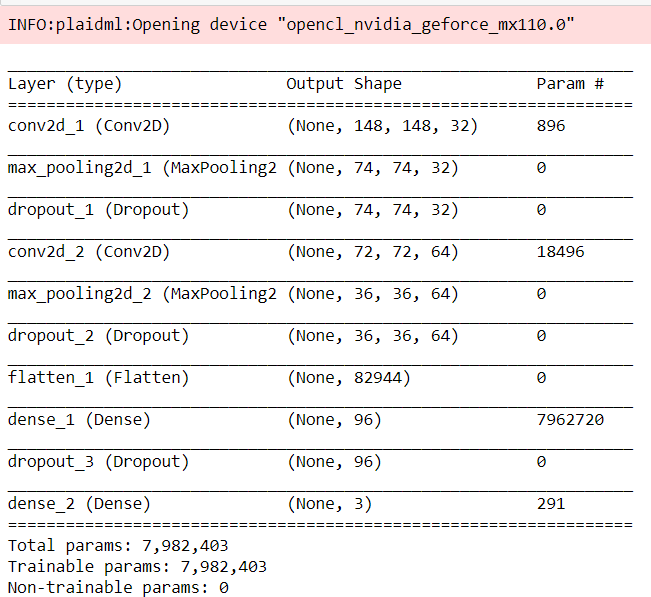
model2.add(Dropout(0.25))

model2.add(Conv2D(64, (3, 3), activation ='relu')) model2.add(MaxPooling2D(pool\_size=(2,2))) model2.add(Dropout(0.25))

model2.add(Flatten()) model2.add(Dense(96, activation ='relu')) model2.add(Dropout(0.5))

model2.add(Dense(3, activation = "softmax"))

model2.compile(loss='categorical\_crossentropy', optimizer='Adam', metrics=['accuracy']) model2.summary()



#### #Train and Validate Dataset

from keras.callbacks import EarlyStopping, ReduceLROnPlateau earlystop = EarlyStopping(patience = 10)

learning\_rate\_reduction = ReduceLROnPlateau(monitor = 'val\_loss', patience = 5, verbose = 1, factor = 0.5, min\_lr = 0.00001)

callbacks = [earlystop, learning\_rate\_reduction]

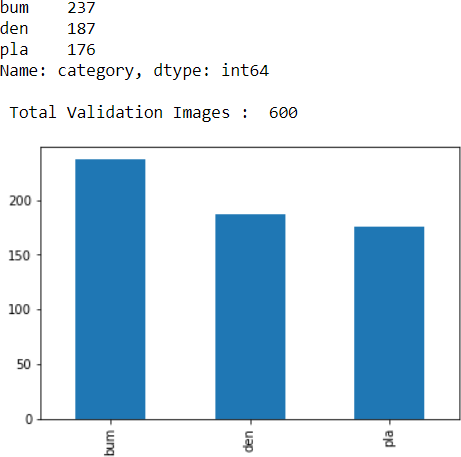
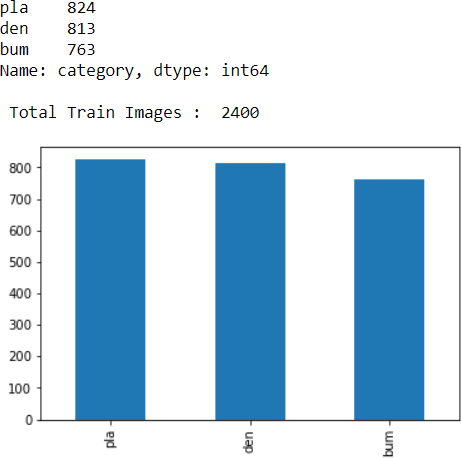
df["category"] = df["category"].replace({0: 'pla', 1: 'den', 2: 'bum'}) df["category"].head()

train\_df, validate\_df = train\_test\_split(df, test\_size = 0.20, random\_state = 0) train\_df = train\_df.reset\_index(drop = True) train\_df['category'].value\_counts().plot.bar() print(train\_df['category'].value\_counts())

print("\n Total Train Images : ", train\_df.shape[0])

validate\_df = validate\_df.reset\_index(drop = True) validate\_df['category'].value\_counts().plot.bar() print(validate\_df['category'].value\_counts())

print("\n Total Validation Images : ", validate\_df.shape[0])



#### #Train and Validate Dataset into CSV file

train\_files\_df = train\_df.copy() validate\_files\_df = validate\_df.copy()

train\_files\_df.to\_csv('train\_files.csv', index = False)

validate\_files\_df.to\_csv('validate\_files.csv', index = False) total\_train = train\_df.shape[0]

print(" Total Train Images : ", total\_train) total\_validate = validate\_df.shape[0]

print(" Total Validate Images : ", total\_validate)

#### #Data Augmentation (Both Training and Validation)

train\_datagen = ImageDataGenerator( rotation\_range = 15, rescale = 1./255,

shear\_range = 0.1,

zoom\_range = 0.2, horizontal\_flip = True, width\_shift\_range = 0.1,

height\_shift\_range = 0.1,

brightness\_range = (0.5, 1.5)) validation\_datagen = ImageDataGenerator(rescale = 1./255)

#### #Training and Validation Generator

train\_generator = train\_datagen.flow\_from\_dataframe( train\_df,

"E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\",

x\_col = 'filename', y\_col = 'category', target\_size = (150,150),

class\_mode = 'categorical', batch\_size = 32

)

validation\_generator = validation\_datagen.flow\_from\_dataframe( validate\_df,

"E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\",

x\_col = 'filename',

y\_col = 'category', target\_size = (150,150), class\_mode = 'categorical', batch\_size = 32

)



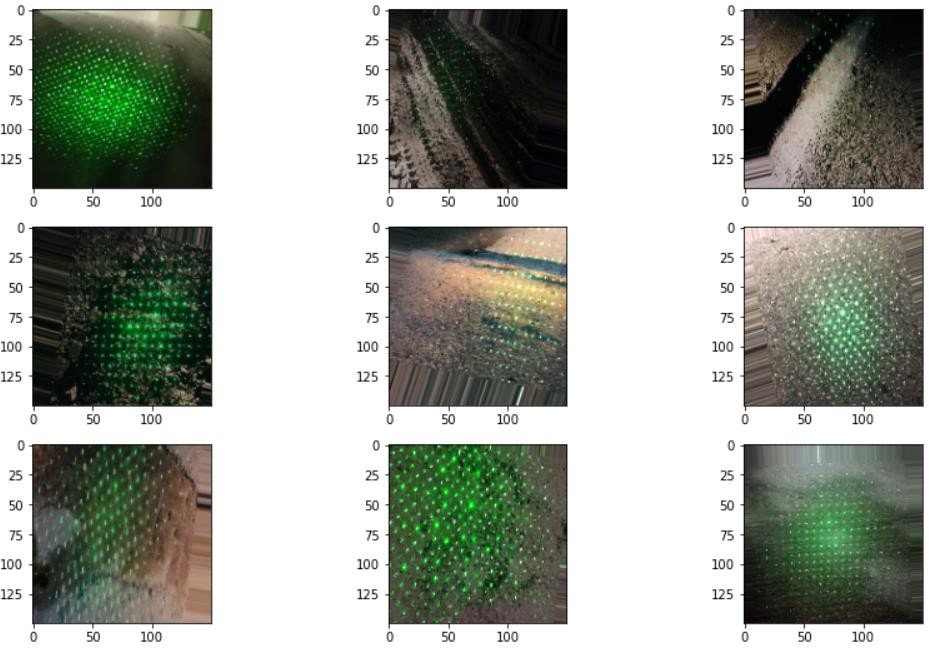
#### #Augmented Image for Training Generator

plt.figure(figsize =(12,12)) for i in range(0,12):

plt.subplot(5, 3, i+1)

for X\_batch, Y\_batch in train\_generator: image = X\_batch[0] plt.imshow(image)

break plt.tight\_layout() plt.show()



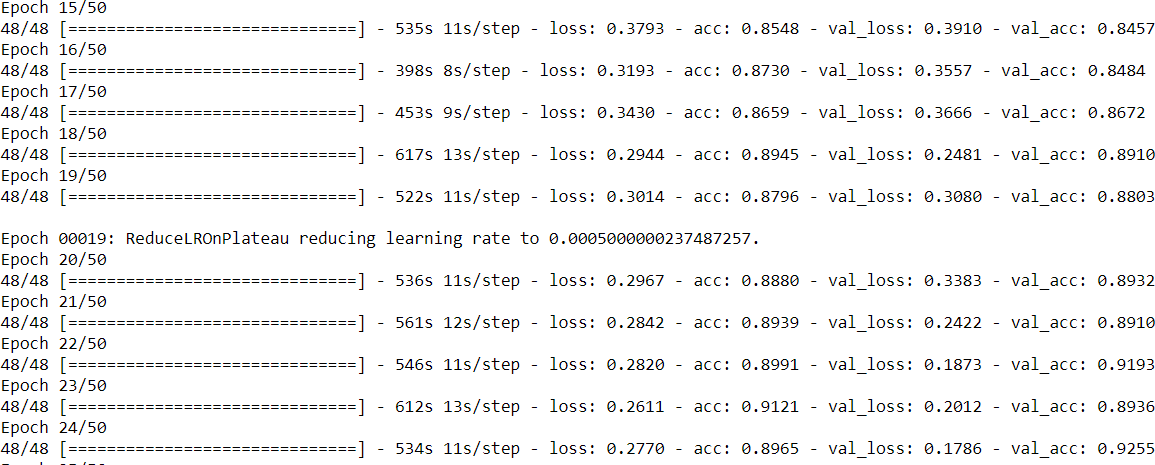
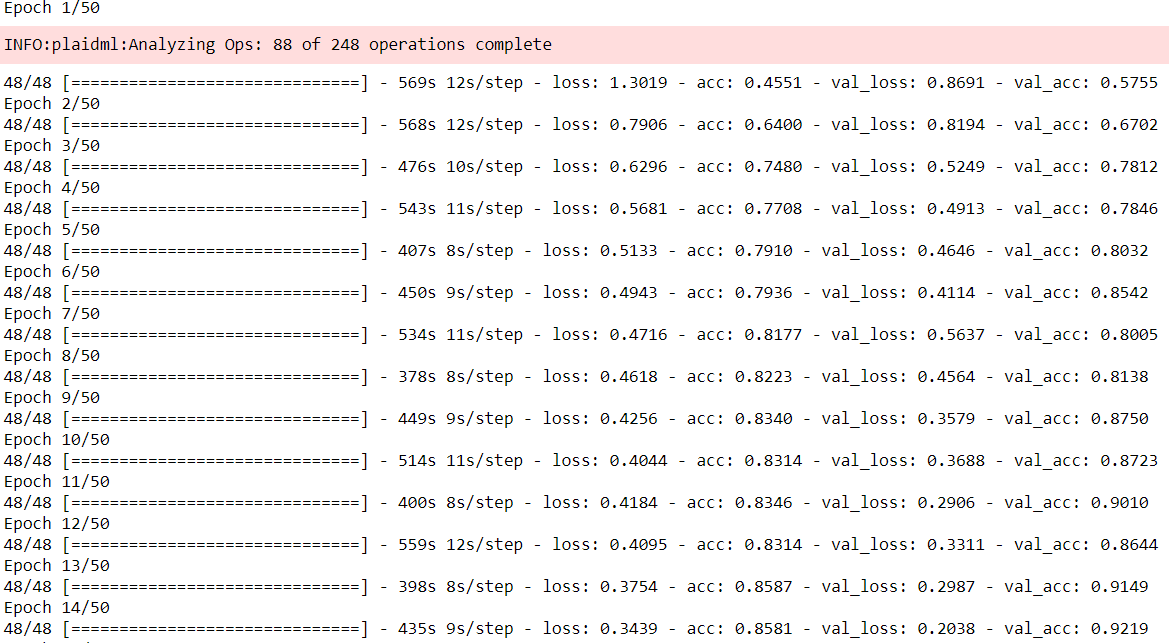
#### #TRAINING THE PROPOSED MODEL

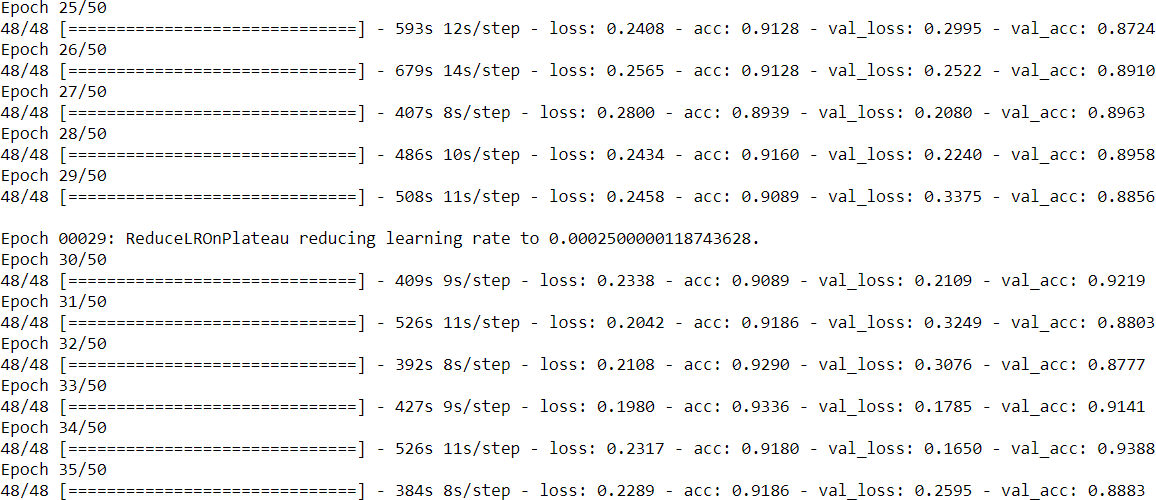
epochs = 50

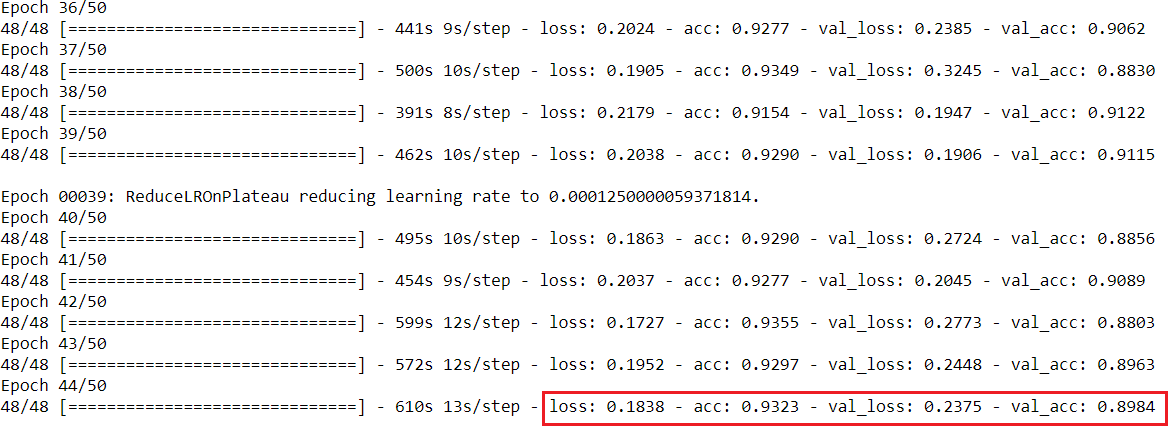
batch\_size = 50

history = model2.fit\_generator( train\_generator,

epochs = epochs,

validation\_data = validation\_generator, validation\_steps = total\_validate/batch\_size, steps\_per\_epoch = total\_train/batch\_size, callbacks = callbacks)





#### #Training Loss vs Validation Loss & Training Accuracy vs Validation Accuracy

fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(6,6)) ax1.plot(history.history['loss'], color ='b', label = "Training Loss") ax1.plot(history.history['val\_loss'], color ='r', label = "Validation Loss") ax1.set\_xticks(np.arange(1, epochs, 1))

ax1.set\_yticks(np.arange(0, 1, 0.1)) ax1.legend()

ax2.plot(history.history['acc'], color ='b', label = "Training Accuracy") ax2.plot(history.history['val\_acc'], color ='r', label = "Validation Accuracy") ax1.set\_xticks(np.arange(1, epochs, 1))

ax2.legend()

legend = plt.legend(loc='best', shadow =True) plt.tight\_layout()

plt.show()



#### #Importing Test Dataset

test\_filenames = os.listdir("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\Test File 3\\Different Test Image Files\\T18")

test\_df = pd.DataFrame({ 'filename': test\_filenames}) total\_test = test\_df.shape[0]

print(" Total Number of Test Images : ", total\_test) **#Augmentation of Test Dataset and Test Generator** test\_datagen = ImageDataGenerator( #rotation\_range = 15,

#rescale = 1./255, #shear\_range = 0.1,

zoom\_range = 0.2, #horizontal\_flip = True, width\_shift\_range = 0.1,

height\_shift\_range = 0.1,

brightness\_range = (0.5, 1.5))

batch\_size = 15

test\_generator = test\_datagen.flow\_from\_dataframe( test\_df,

"E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\Test File 3\\Different Test Image Files\\T18",

x\_col = 'filename', y\_col = None,

class\_mode = None, target\_size = (150,150), batch\_size =batch\_size, shuffle = False

)

#### #Prediction of Test Data Array Wise

predict = model2.predict\_generator(test\_generator, steps=np.ceil(total\_test/batch\_size)) print(" Number of Predicted Images : ", len(predict), "\n")

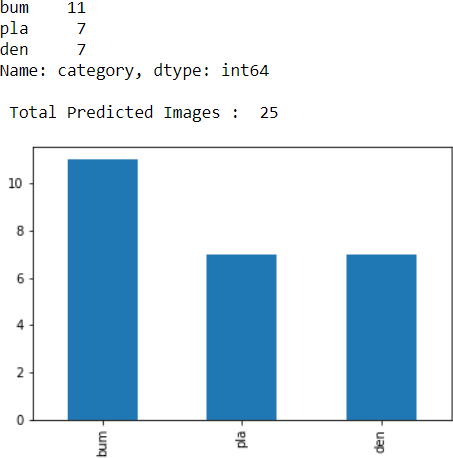
print (" 0:pla 1:den 2:bum \n\n", predict, "\n") **#Prediction Bar Chart Using Test Images** test\_df['category'].value\_counts() print(test\_df['category'].value\_counts())

##### #plot for shpwing bar graph

test\_df['category'].value\_counts().plot.bar()

##### #Total Predicted Images

print("\n Total Predicted Images : ", test\_df.shape[0])



**#Prediction Using Test Images** sample\_test = test\_df.head(25) sample\_test.head() plt.figure(figsize=(12,24))

for index, row in sample\_test.iterrows(): filename = row['filename']

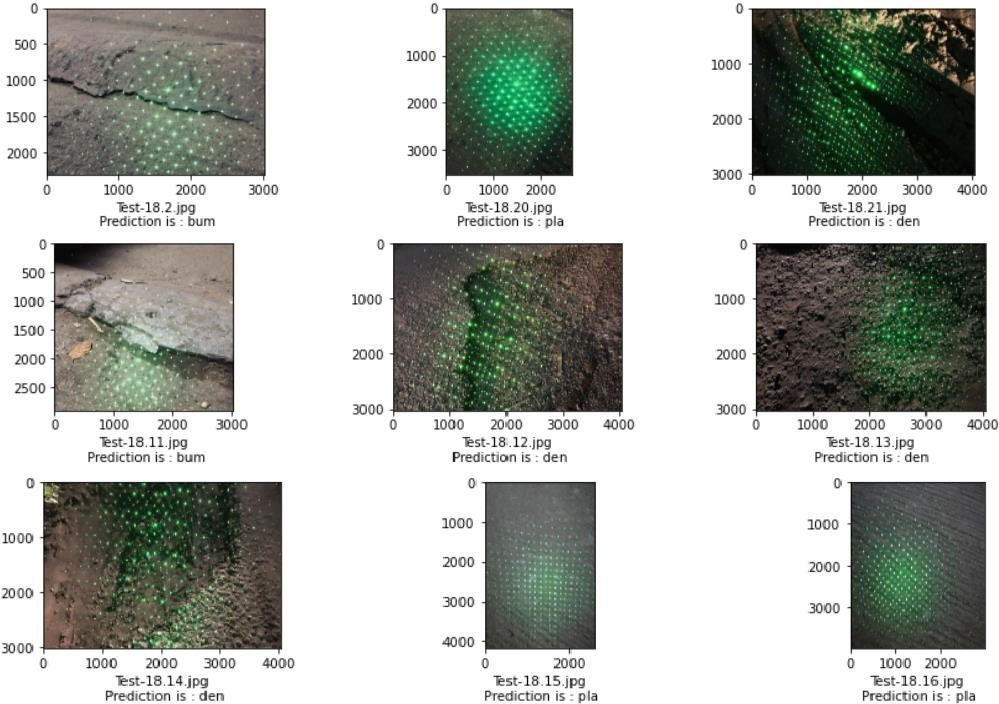
category = row['category']

img = load\_img("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\Test File 3\\Different Test Image Files\\T18\\"+filename)

plt.subplot(9, 3, index+1) plt.imshow(img)

plt.xlabel(filename+ "\n Prediction is : " + "{}".format(category)) plt.tight\_layout()

plt.show()

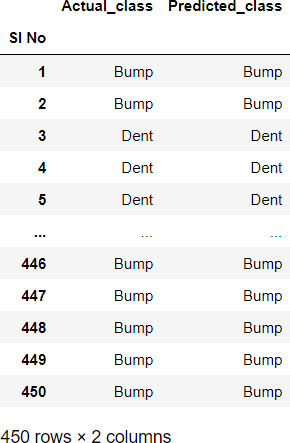


#### #Code for Confusion Matrix

import pandas as pd import seaborn as sn

import matplotlib.pyplot as plt import numpy as np

pd.read\_excel("Main\_Code\_Model\_Output (Most Accurate 450 Test Image).xlsx", index\_col =[0])



create\_confusion\_matrix= pd.crosstab(Model\_pred.Predicted\_class, Model\_pred.Actual\_class) fig = plt.figure(figsize = (17,5))

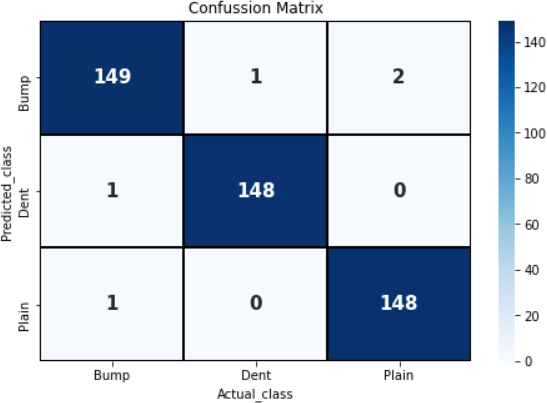
ax = plt.subplot(121)

sn.heatmap(create\_confusion\_matrix, annot = True, cmap ='Blues',fmt=".0f", linewidth=1, linecolor='black',

annot\_kws={'fontsize':15, 'fontweight': 'bold'}

)

ax.set\_title("Confussion Matrix")



#### #True Positive, True Negative, False Positive, False Negative

##### #True Positive

TP = create\_confusion\_matrix.iloc[1,1]

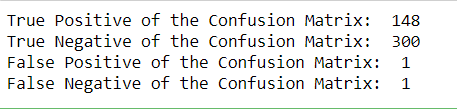
print("True Positive of the Confusion Matrix: ",TP)

##### #True Negative

TN = create\_confusion\_matrix.sum().sum()-TP-FP-FN print("True Negative of the Confusion Matrix: ",TN) **#False Positive**

FP = create\_confusion\_matrix.iloc[1,:].sum()-TP print("False Positive of the Confusion Matrix: ",FP) **#False Negative**

FN = create\_confusion\_matrix.iloc[:,1].sum()-TP print("False Negative of the Confusion Matrix: ",FN)

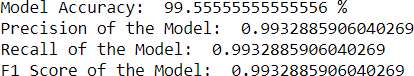


**#Accuracy, Precision, Recall, F1 Score of the Model** Accuracy = (TP+TN)/create\_confusion\_matrix.sum().sum() print ("Model Accuracy: ",Accuracy\*100,"%")

Precision = TP/(TP+FP)

print("Precision of the Model: ",Precision) Recall = TP/(TP+FN)

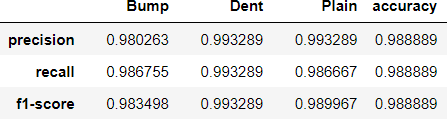
print("Recall of the Model: ",Recall)

F1\_Score = (2 \*Precision \*Recall)/(Precision+Recall) print("F1 Score of the Model: ",F1\_Score)

#### #Accuracy, Precision, Recall, F1 Score for Individual Plane, Dent, Bump Road

from sklearn.metrics import classification\_report

pd.DataFrame(classification\_report(Model\_pred.Actual\_class,Model\_pred.Predicted\_class,output\_di ct =True))



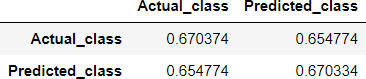
#### #Covarience, Correlation Coefficient (Took Plain=1, Dent=2, Bump=3)

fahim =

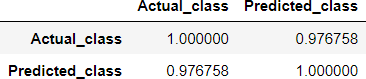
pd.read\_excel("RMSE\_CORR\_COV (Most Accurate 450 Test Image).xlsx",index\_col

=[0])

fahim.cov()



fahim.corr(method="pearson")



#### #Calculate MSE, RMSE, MAE, MAPE, R2 Score (Coefficient of Determination)

from sklearn import metrics

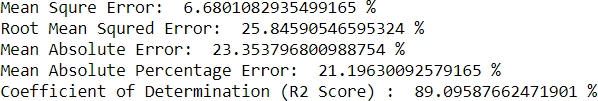
from sklearn.metrics import mean\_squared\_error from sklearn import linear\_model

mse = mean\_squared\_error(y\_test,y\_pred\_LR)

rmse = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_LR)) mae = np.sqrt(metrics.mean\_absolute\_error(y\_test, y\_pred\_LR))

mape = np.sqrt(metrics.mean\_absolute\_percentage\_error(y\_test, y\_pred\_LR)) R2 = metrics.r2\_score(y\_test, y\_pred\_LR)

print("Mean Squre Error: ",mse\*100,"%") print("Root Mean Squred Error: ",rmse\*100,"%") print("Mean Absolute Error: ",mae\*100,"%")

print("Mean Absolute Percentage Error: ",mape\*100,"%") print("Coefficient of Determination (R2 Score) : ",R2\*100,"%")

**Contour Detection, Edge Detection, Histogram Equalization**

**#Contour on Dent and Bump Road**

img = cv2.imread('E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\3000 Image Folder Category Wise\\Train Images\\Dent 1000\\den.547.jpg')

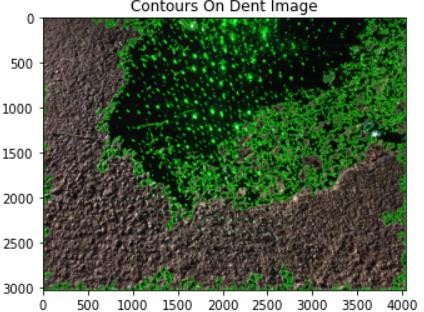
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) #plt.figure(figsize=[9,9])

lower = np.array([30,30,30]) higher = np.array([250,250,250])

mask = cv2.inRange(img,lower,higher)

cont,\_ = cv2.findContours(mask,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE) cont\_img = cv2.drawContours(img,cont,-1, (0, 255, 0), 2)

plt.figure(figsize=[8,8]); plt.imshow(cont\_img); plt.title("Contours On Bump Image")



img = cv2.imread('E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\3000 Image Folder Category Wise\\Train Images\\Bump 1000\\bum.31.jpg')

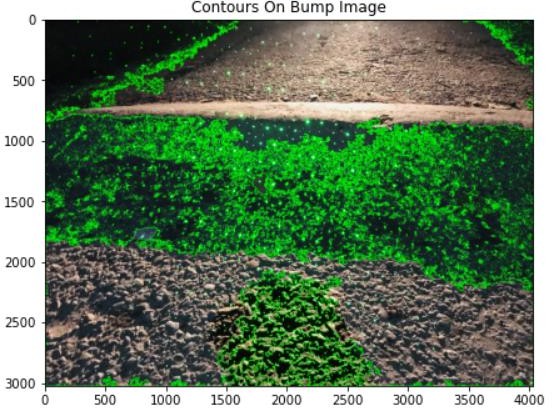
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) #plt.figure(figsize=[9,9])

lower = np.array([30,30,30]) higher = np.array([250,250,250])

mask = cv2.inRange(img,lower,higher)

cont,\_ = cv2.findContours(mask,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE) cont\_img = cv2.drawContours(img,cont,-1, (0, 255, 0), 2)

plt.figure(figsize=[8,8]); plt.imshow(cont\_img); plt.title("Contours On Bump Image")



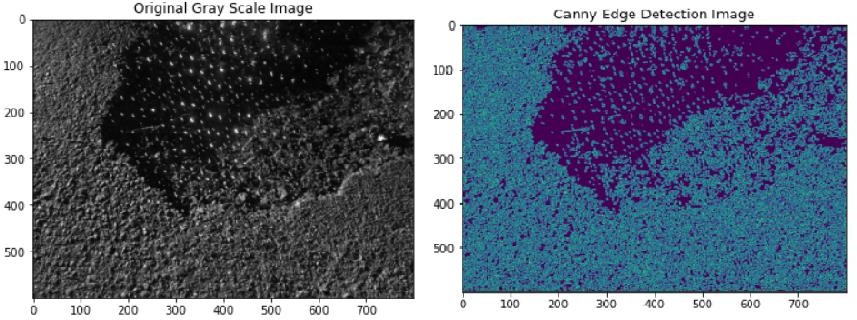
#### #Canny Edge Detection on Dent Road

img = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\den.547.jpg",0)

img = cv2.resize(img, (800,600)) CannyEdgeDetection = cv2.Canny(img, 100, 200) plt.figure(figsize=[6,6])

plt.title("Original Gray Scale Image"); plt.imshow(img,'gray')

plt.figure(figsize=[6,6]); plt.title("Canny Edge Detection Image"); plt.imshow(CannyEdgeDetection)



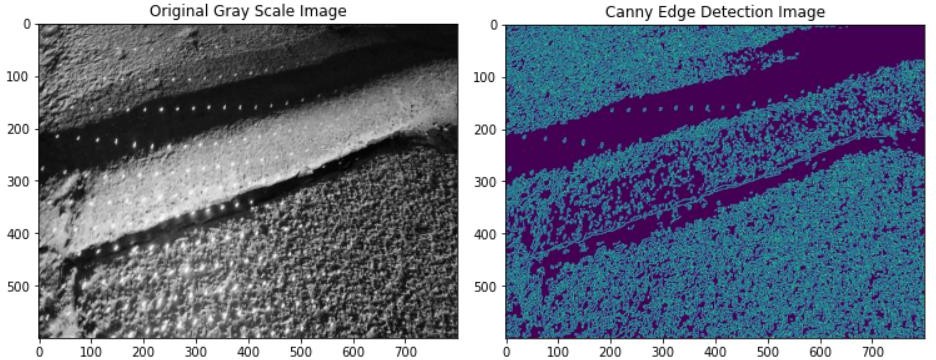
#### #Canny Edge Detection on Bump Road

img = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\bum.474.jpg",0)

img = cv2.resize(img, (800,600)) CannyEdgeDetection = cv2.Canny(img, 100, 200) plt.figure(figsize=[6,6])

plt.title("Original Gray Scale Image"); plt.imshow(img,'gray')

plt.figure(figsize=[6,6]); plt.title("Canny Edge Detection Image"); plt.imshow(CannyEdgeDetection)



#### #Histogram Applied on Plane, Dent and Bump Road Together

import cv2

from matplotlib import pyplot as plt

imgP = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\pla.624.jpg")

imgD = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\den.547.jpg")

imgB = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\bum.474.jpg")

imgP= cv2.resize(imgP, (800,600))

imgD= cv2.resize(imgD, (800,600))

imgB= cv2.resize(imgB, (800,600))

cv2.imshow("Plane Road Image", imgP) cv2.imshow("Dent Road Image", imgD) cv2.imshow("Bump Road Image", imgB)

plt.hist(imgP.ravel(), 256, [0, 256], color = 'blue')

plt.hist(imgD.ravel(), 256, [0, 256],color = 'red')

plt.hist(imgB.ravel(), 256, [0, 256],color = 'green') plt.xlabel("Intensity Level")

plt.ylabel("Intensity Frequency")

plt.title("Histogram of Plane, Dent and Bump Road")

plt.legend(['Plain Road', 'Dent Road', 'Bump Road']) plt.show()

cv2.waitKey(0) cv2.destroyAllWindows()

histP,binsP = np.histogram(imgP.flatten(),256,[0,256]) cdfP = histP.cumsum()

cdfP\_normalized = cdfP \* float(histP.max()) / cdfP.max()

histD,binsD = np.histogram(imgD.flatten(),256,[0,256]) cdfD = histD.cumsum()

cdfD\_normalized = cdfD \* float(histD.max()) / cdfD.max()

histB,binsB = np.histogram(imgB.flatten(),256,[0,256]) cdfB = histB.cumsum()

cdfB\_normalized = cdfB \* float(histB.max()) / cdfB.max()

plt.plot(cdfP\_normalized, color = 'blue') plt.plot(cdfD\_normalized, color = 'red') plt.plot(cdfB\_normalized, color = 'green')

plt.hist(imgP.flatten(),256,[0,256], color = 'blue') plt.hist(imgD.flatten(),256,[0,256], color = 'red') plt.hist(imgB.flatten(),256,[0,256], color = 'green')

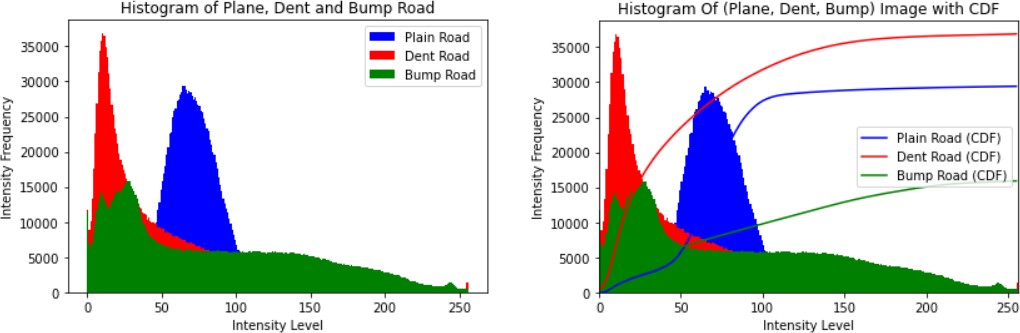
plt.xlim([0,256])

plt.legend(['Plain Road (CDF)', 'Dent Road (CDF)', 'Bump Road (CDF)'], loc = 'center right') plt.xlabel("Intensity Level")

plt.ylabel("Intensity Frequency")

#plt.legend(('CDF(Cumulative Distributive Function)','Histogram'), loc = 'center right') plt.title("Histogram Of (Plane, Dent, Bump) Image with CDF")

plt.show()



#### #Histogram Equalization Applied on Bump Road Image

img = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\bum.31.jpg",1)

img = cv2.resize(img, (800,600))

cv2.imshow('Original Bump Image',img) cv2.waitKey(0) cv2.destroyAllWindows()

hist,bins = np.histogram(img.flatten(),256,[0,256]) cdf = hist.cumsum()

cdf\_normalized = cdf \* float(hist.max()) / cdf.max() plt.plot(cdf\_normalized, color = 'blue') plt.hist(img.flatten(),256,[0,256], color = 'red') plt.xlim([0,256])

plt.legend(('CDF(Cumulative Distributive Function)','Histogram'), loc = 'center right') plt.title("Histogram Of the Bump Image")

plt.show()

img = cv2.imread("E:\\AIUB\\12th Semester\\OFFICIAL THESIS\\CODE\\All Image Files For New Code (3000 Images)\\plain\_dent\_bump\_3\\data\\bum.31.jpg",0)

img = cv2.resize(img, (800,600))

**#This is the Histogram Equalization Code** sakib = cv2.equalizeHist(img) cv2.imshow('Histogram Equalised Image',sakib)

cv2.waitKey(0) cv2.destroyAllWindows()

hist,bins = np.histogram(sakib.flatten(),256,[0,256]) cdf = hist.cumsum()

cdf\_normalized = cdf \* float(hist.max()) / cdf.max() plt.plot(cdf\_normalized, color = 'b') plt.hist(sakib.flatten(),256,[0,256], color = 'r') plt.xlim([0,256])

plt.legend(('CDF(Cumulative Distributive Function)','Histogram'), loc = 'upper right') plt.title("Histogram Equalised Of the Bump Image")

plt.show()

