In [ ]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import os
import pickle
import matplotlib.patches as patches
import re
import random
from sklearn.model_selection import train_test_split
import cv2
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import plot_model
from PIL import Image
import tensorflow as tf
from keras.layers import GlobalAveragePooling2D,Dense,Conv2D,BatchNormalization,Dropout
import keras
from keras import backend as K
from keras.models import Model,load_model
from tensorflow.python.framework.ops import disable_eager_execution
from keras.regularizers import l2
import datetime
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [ ]:
```
with open('/content/drive//My Drive/Steel_Detection /data.pkl','rb') as f:
    train=pickle.load(f)
```

In [ ]:
```
train.head()
```

	image_id	rle_1	rle_2	rle_3	rle_4	defect	stratify	defect_1	defect_2	defect_3	defect_4	total_defects
0	0002cc93b.jpg	29102 12 29346 24 29602 24 29858 24 30114 24 3...				1	1	1	0	0	0	1
1	00031f466.jpg					0	0	0	0	0	0	0
2	000418bfc.jpg					0	0	0	0	0	0	0
3	000789191.jpg					0	0	0	0	0	0	0
4	0007a71bf.jpg			18661 28 18863 82 19091 110 19347 110 19603 11...		1	3	0	0	1	0	1

In [ ]:
```
train.shape
```

```
(12568, 12)
```

In [ ]:
```
def f1_score(y_true, y_pred):
    #https://aakashgoel12.medium.com/how-to-add-user-defined-function-get-f1-score-in-keras-metrics-3013f979ce0d
    #https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras
    true_positives=K.sum(K.round(K.clip(y_true*y_pred,0,1)))   #calculates number of true positives
    possible_positives=K.sum(K.round(K.clip(y_true,0,1)))      #calculates number of actual positives
    predicted_positives=K.sum(K.round(K.clip(y_pred,0,1)))

    #K.epsilon takes care of non-zero divisions
    #was modified by adding the constant epsilon, in order to avoid division by 0. Thus NaN will not be computed.
    precision=true_positives/(predicted_positives +K.epsilon())
    recall=true_positives/(possible_positives+K.epsilon())
    f1_val=2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

In [ ]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
x_train,x_test=train_test_split(train,test_size=0.10,stratify=train['stratify'],random_state=0)
x_train,x_val=train_test_split(x_train,test_size=0.20,stratify=x_train['stratify'],random_state=0)
x_train=x_train[['image_id','defect']]
x_val=x_val[['image_id','defect']]
x_test=x_test[['image_id','defect']]
print("x_train {}".format(x_train.shape),"  x_val {}".format(x_val.shape)," x_test {}".format(x_test.shape))
```

```
x_train (9048, 2)   x_val (2263, 2)  x_test (1257, 2)
```

- Data augmentation is used here to increase the amount of data by adding slightly modified copies of already existing data also during EDA we have seen that data is highly imbalanced. So it is advisable to do at least Data augmentation.

In [ ]:
```
#https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatag
enerator/
#https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,
                                 shear_range=0.1,
                                 zoom_range=0.1,
                                 horizontal_flip=True,
                                 vertical_flip=True,
                                 rotation_range=60)

val_datagen=ImageDataGenerator(rescale=1./255)


train_folder_path='/content/drive//My Drive/Steel_Detection /train_images'

train_image_generator=train_datagen.flow_from_dataframe(dataframe=x_train.astype(str),
                                                        directory=train_folder_path,
                                                        x_col="image_id",
                                                        y_col="defect",
                                                        seed=42,
                                                        shuffle=True,
                                                        batch_size=32,
                                                        class_mode="binary",
                                                        target_size=(256,512))

val_image_generator=val_datagen.flow_from_dataframe(dataframe=x_val.astype(str),
                                                        directory=train_folder_path,
                                                        x_col="image_id",
                                                        y_col="defect",
                                                        batch_size=32,
                                                        seed=42,
                                                        shuffle=True,
```

```
    class_mode="binary",
    target_size=(256,512))
```

```
Found 9048 validated image filenames belonging to 2 classes.
Found 2263 validated image filenames belonging to 2 classes.
```

- Flattening is No brainer and it simply converts a multi-dimensional object to one-dimensional by re-arranging the elements.
- While GlobalAveragePooling is a methodology used for better representation of your vector. It can be 1D/2D/3D.It uses a parser window which moves across the object and pools the data by averaging it (GlobalAveragePooling) or picking max value (GlobalMaxPooling).

- Batch normalization, it is a process to make neural networks faster and more stable through adding extra layers in a deep neural network.The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.A typical neural network is trained using a collected set of input data called batch. Similarly, the normalizing process in batch normalization takes place in batches, not as a single input.

- Large neural nets trained on relatively small datasets can overfit the training data.By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections we can overcome overfitting problem.

- Used pre-trained Xception() model by keras without fully-connected layer at the top of the network and later freeze the pre-trained model.

In [ ]:

```
#https://keras.io/api/applications/xception/
base_model=tf.keras.applications.xception.Xception(input_shape=(256,512,3),include_top=False)
base_model.trainable=False

m=base_model.output
# add a global average pooling layer
#https://stackoverflow.com/questions/49295311/what-is-the-difference-between-flatten-and-globalaveragepooling2d-in-keras
m=GlobalAveragePooling2D()(m)

# add fully-connected layers
m=Dense(1024,activation='relu')(m)
m=BatchNormalization()(m)   #https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/
m=Dropout(0.3)(m)

#https://machinelearningmastery.com/how-to-reduce-overfitting-in-deep-learning-with-weight-regularization/
m=Dense(512,activation='relu')(m)
m=BatchNormalization()(m)
m= Dropout(0.3)(m)

m=Dense(64,activation='relu')(m)
m=BatchNormalization()(m)
m=Dropout(0.3)(m)

#prediction layer
output=Dense(1,activation='sigmoid')(m)   #Binary Classification thus sigmoid is used

model=Model(inputs=base_model.input,outputs=output)
model._name="Binary_Classification_Model"
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_no
top.h5
83689472/83683744 [==============================] - 1s 0us/step
83697664/83683744 [==============================] - 1s 0us/step
Model: "Binary_Classification_Model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 512, 3)	0	
block1_conv1 (Conv2D)	(None, 127, 255, 32)	864	input_1[0][0]
block1_conv1_bn (BatchNormaliza	(None, 127, 255, 32)	128	block1_conv1[0][0]
block1_conv1_act (Activation)	(None, 127, 255, 32)	0	block1_conv1_bn[0][0]
block1_conv2 (Conv2D)	(None, 125, 253, 64)	18432	block1_conv1_act[0][0]
block1_conv2_bn (BatchNormaliza	(None, 125, 253, 64)	256	block1_conv2[0][0]
block1_conv2_act (Activation)	(None, 125, 253, 64)	0	block1_conv2_bn[0][0]
block2_sepconv1 (SeparableConv2	(None, 125, 253, 128	8768	block1_conv2_act[0][0]
block2_sepconv1_bn (BatchNormal	(None, 125, 253, 128	512	block2_sepconv1[0][0]
block2_sepconv2_act (Activation	(None, 125, 253, 128	0	block2_sepconv1_bn[0][0]
block2_sepconv2 (SeparableConv2	(None, 125, 253, 128	17536	block2_sepconv2_act[0][0]
block2_sepconv2_bn (BatchNormal	(None, 125, 253, 128	512	block2_sepconv2[0][0]
conv2d (Conv2D)	(None, 63, 127, 128)	8192	block1_conv2_act[0][0]
block2_pool (MaxPooling2D)	(None, 63, 127, 128)	0	block2_sepconv2_bn[0][0]
batch_normalization (BatchNorma	(None, 63, 127, 128)	512	conv2d[0][0]
add (Add)	(None, 63, 127, 128)	0	block2_pool[0][0] batch_normalization[0][0]
block3_sepconv1_act (Activation	(None, 63, 127, 128)	0	add[0][0]

block3_sepconv1 (SeparableConv2	(None, 63, 127, 256)	33920	block3_sepconv1_act[0][0]	
block3_sepconv1_bn (BatchNormal	(None, 63, 127, 256)	1024	block3_sepconv1[0][0]	
block3_sepconv2_act (Activation	(None, 63, 127, 256)	0	block3_sepconv1_bn[0][0]	
block3_sepconv2 (SeparableConv2	(None, 63, 127, 256)	67840	block3_sepconv2_act[0][0]	
block3_sepconv2_bn (BatchNormal	(None, 63, 127, 256)	1024	block3_sepconv2[0][0]	
conv2d_1 (Conv2D)	(None, 32, 64, 256)	32768	add[0][0]	
block3_pool (MaxPooling2D)	(None, 32, 64, 256)	0	block3_sepconv2_bn[0][0]	
batch_normalization_1 (BatchNor	(None, 32, 64, 256)	1024	conv2d_1[0][0]	
add_1 (Add)	(None, 32, 64, 256)	0	block3_pool[0][0] batch_normalization_1[0][0]	
block4_sepconv1_act (Activation	(None, 32, 64, 256)	0	add_1[0][0]	
block4_sepconv1 (SeparableConv2	(None, 32, 64, 728)	188672	block4_sepconv1_act[0][0]	
block4_sepconv1_bn (BatchNormal	(None, 32, 64, 728)	2912	block4_sepconv1[0][0]	
block4_sepconv2_act (Activation	(None, 32, 64, 728)	0	block4_sepconv1_bn[0][0]	
block4_sepconv2 (SeparableConv2	(None, 32, 64, 728)	536536	block4_sepconv2_act[0][0]	
block4_sepconv2_bn (BatchNormal	(None, 32, 64, 728)	2912	block4_sepconv2[0][0]	
conv2d_2 (Conv2D)	(None, 16, 32, 728)	186368	add_1[0][0]	
block4_pool (MaxPooling2D)	(None, 16, 32, 728)	0	block4_sepconv2_bn[0][0]	
batch_normalization_2 (BatchNor	(None, 16, 32, 728)	2912	conv2d_2[0][0]	
add_2 (Add)	(None, 16, 32, 728)	0	block4_pool[0][0] batch_normalization_2[0][0]	
block5_sepconv1_act (Activation	(None, 16, 32, 728)	0	add_2[0][0]	
block5_sepconv1 (SeparableConv2	(None, 16, 32, 728)	536536	block5_sepconv1_act[0][0]	

block5_sepconv1_bn (BatchNormal	(None, 16, 32, 728)	2912	block5_sepconv1[0][0]	
block5_sepconv2_act (Activation	(None, 16, 32, 728)	0	block5_sepconv1_bn[0][0]	
block5_sepconv2 (SeparableConv2	(None, 16, 32, 728)	536536	block5_sepconv2_act[0][0]	
block5_sepconv2_bn (BatchNormal	(None, 16, 32, 728)	2912	block5_sepconv2[0][0]	
block5_sepconv3_act (Activation	(None, 16, 32, 728)	0	block5_sepconv2_bn[0][0]	
block5_sepconv3 (SeparableConv2	(None, 16, 32, 728)	536536	block5_sepconv3_act[0][0]	
block5_sepconv3_bn (BatchNormal	(None, 16, 32, 728)	2912	block5_sepconv3[0][0]	
add_3 (Add)	(None, 16, 32, 728)	0	block5_sepconv3_bn[0][0] add_2[0][0]	
block6_sepconv1_act (Activation	(None, 16, 32, 728)	0	add_3[0][0]	
block6_sepconv1 (SeparableConv2	(None, 16, 32, 728)	536536	block6_sepconv1_act[0][0]	
block6_sepconv1_bn (BatchNormal	(None, 16, 32, 728)	2912	block6_sepconv1[0][0]	
block6_sepconv2_act (Activation	(None, 16, 32, 728)	0	block6_sepconv1_bn[0][0]	
block6_sepconv2 (SeparableConv2	(None, 16, 32, 728)	536536	block6_sepconv2_act[0][0]	
block6_sepconv2_bn (BatchNormal	(None, 16, 32, 728)	2912	block6_sepconv2[0][0]	
block6_sepconv3_act (Activation	(None, 16, 32, 728)	0	block6_sepconv2_bn[0][0]	
block6_sepconv3 (SeparableConv2	(None, 16, 32, 728)	536536	block6_sepconv3_act[0][0]	
block6_sepconv3_bn (BatchNormal	(None, 16, 32, 728)	2912	block6_sepconv3[0][0]	
add_4 (Add)	(None, 16, 32, 728)	0	block6_sepconv3_bn[0][0] add_3[0][0]	
block7_sepconv1_act (Activation	(None, 16, 32, 728)	0	add_4[0][0]	
block7_sepconv1 (SeparableConv2	(None, 16, 32, 728)	536536	block7_sepconv1_act[0][0]	

block7_sepconv1_bn (BatchNormal	(None, 16, 32, 728)	2912	block7_sepconv1[0][0]	
block7_sepconv2_act (Activation	(None, 16, 32, 728)	0	block7_sepconv1_bn[0][0]	
block7_sepconv2 (SeparableConv2	(None, 16, 32, 728)	536536	block7_sepconv2_act[0][0]	
block7_sepconv2_bn (BatchNormal	(None, 16, 32, 728)	2912	block7_sepconv2[0][0]	
block7_sepconv3_act (Activation	(None, 16, 32, 728)	0	block7_sepconv2_bn[0][0]	
block7_sepconv3 (SeparableConv2	(None, 16, 32, 728)	536536	block7_sepconv3_act[0][0]	
block7_sepconv3_bn (BatchNormal	(None, 16, 32, 728)	2912	block7_sepconv3[0][0]	
add_5 (Add)	(None, 16, 32, 728)	0	block7_sepconv3_bn[0][0] add_4[0][0]	
block8_sepconv1_act (Activation	(None, 16, 32, 728)	0	add_5[0][0]	
block8_sepconv1 (SeparableConv2	(None, 16, 32, 728)	536536	block8_sepconv1_act[0][0]	
block8_sepconv1_bn (BatchNormal	(None, 16, 32, 728)	2912	block8_sepconv1[0][0]	
block8_sepconv2_act (Activation	(None, 16, 32, 728)	0	block8_sepconv1_bn[0][0]	
block8_sepconv2 (SeparableConv2	(None, 16, 32, 728)	536536	block8_sepconv2_act[0][0]	
block8_sepconv2_bn (BatchNormal	(None, 16, 32, 728)	2912	block8_sepconv2[0][0]	
block8_sepconv3_act (Activation	(None, 16, 32, 728)	0	block8_sepconv2_bn[0][0]	
block8_sepconv3 (SeparableConv2	(None, 16, 32, 728)	536536	block8_sepconv3_act[0][0]	
block8_sepconv3_bn (BatchNormal	(None, 16, 32, 728)	2912	block8_sepconv3[0][0]	
add_6 (Add)	(None, 16, 32, 728)	0	block8_sepconv3_bn[0][0] add_5[0][0]	
block9_sepconv1_act (Activation	(None, 16, 32, 728)	0	add_6[0][0]	
block9_sepconv1 (SeparableConv2	(None, 16, 32, 728)	536536	block9_sepconv1_act[0][0]	
block9_sepconv1_bn (BatchNormal	(None, 16, 32, 728)	2912	block9_sepconv1[0][0]	

block9_sepconv2_act (Activation	(None, 16, 32, 728)	0	block9_sepconv1_bn[0][0]
block9_sepconv2 (SeparableConv2	(None, 16, 32, 728)	536536	block9_sepconv2_act[0][0]
block9_sepconv2_bn (BatchNormal	(None, 16, 32, 728)	2912	block9_sepconv2[0][0]
block9_sepconv3_act (Activation	(None, 16, 32, 728)	0	block9_sepconv2_bn[0][0]
block9_sepconv3 (SeparableConv2	(None, 16, 32, 728)	536536	block9_sepconv3_act[0][0]
block9_sepconv3_bn (BatchNormal	(None, 16, 32, 728)	2912	block9_sepconv3[0][0]
add_7 (Add)	(None, 16, 32, 728)	0	block9_sepconv3_bn[0][0] add_6[0][0]
block10_sepconv1_act (Activatio	(None, 16, 32, 728)	0	add_7[0][0]
block10_sepconv1 (SeparableConv	(None, 16, 32, 728)	536536	block10_sepconv1_act[0][0]
block10_sepconv1_bn (BatchNorma	(None, 16, 32, 728)	2912	block10_sepconv1[0][0]
block10_sepconv2_act (Activatio	(None, 16, 32, 728)	0	block10_sepconv1_bn[0][0]
block10_sepconv2 (SeparableConv	(None, 16, 32, 728)	536536	block10_sepconv2_act[0][0]
block10_sepconv2_bn (BatchNorma	(None, 16, 32, 728)	2912	block10_sepconv2[0][0]
block10_sepconv3_act (Activatio	(None, 16, 32, 728)	0	block10_sepconv2_bn[0][0]
block10_sepconv3 (SeparableConv	(None, 16, 32, 728)	536536	block10_sepconv3_act[0][0]
block10_sepconv3_bn (BatchNorma	(None, 16, 32, 728)	2912	block10_sepconv3[0][0]
add_8 (Add)	(None, 16, 32, 728)	0	block10_sepconv3_bn[0][0] add_7[0][0]
block11_sepconv1_act (Activatio	(None, 16, 32, 728)	0	add_8[0][0]
block11_sepconv1 (SeparableConv	(None, 16, 32, 728)	536536	block11_sepconv1_act[0][0]
block11_sepconv1_bn (BatchNorma	(None, 16, 32, 728)	2912	block11_sepconv1[0][0]

block11_sepconv2_act (Activatio	(None, 16, 32, 728)	0	block11_sepconv1_bn[0][0]
block11_sepconv2 (SeparableConv	(None, 16, 32, 728)	536536	block11_sepconv2_act[0][0]
block11_sepconv2_bn (BatchNorma	(None, 16, 32, 728)	2912	block11_sepconv2[0][0]
block11_sepconv3_act (Activatio	(None, 16, 32, 728)	0	block11_sepconv2_bn[0][0]
block11_sepconv3 (SeparableConv	(None, 16, 32, 728)	536536	block11_sepconv3_act[0][0]
block11_sepconv3_bn (BatchNorma	(None, 16, 32, 728)	2912	block11_sepconv3[0][0]
add_9 (Add)	(None, 16, 32, 728)	0	block11_sepconv3_bn[0][0] add_8[0][0]
block12_sepconv1_act (Activatio	(None, 16, 32, 728)	0	add_9[0][0]
block12_sepconv1 (SeparableConv	(None, 16, 32, 728)	536536	block12_sepconv1_act[0][0]
block12_sepconv1_bn (BatchNorma	(None, 16, 32, 728)	2912	block12_sepconv1[0][0]
block12_sepconv2_act (Activatio	(None, 16, 32, 728)	0	block12_sepconv1_bn[0][0]
block12_sepconv2 (SeparableConv	(None, 16, 32, 728)	536536	block12_sepconv2_act[0][0]
block12_sepconv2_bn (BatchNorma	(None, 16, 32, 728)	2912	block12_sepconv2[0][0]
block12_sepconv3_act (Activatio	(None, 16, 32, 728)	0	block12_sepconv2_bn[0][0]
block12_sepconv3 (SeparableConv	(None, 16, 32, 728)	536536	block12_sepconv3_act[0][0]
block12_sepconv3_bn (BatchNorma	(None, 16, 32, 728)	2912	block12_sepconv3[0][0]
add_10 (Add)	(None, 16, 32, 728)	0	block12_sepconv3_bn[0][0] add_9[0][0]
block13_sepconv1_act (Activatio	(None, 16, 32, 728)	0	add_10[0][0]
block13_sepconv1 (SeparableConv	(None, 16, 32, 728)	536536	block13_sepconv1_act[0][0]
block13_sepconv1_bn (BatchNorma	(None, 16, 32, 728)	2912	block13_sepconv1[0][0]
block13_sepconv2_act (Activatio	(None, 16, 32, 728)	0	block13_sepconv1_bn[0][0]

block13_sepconv2 (SeparableConv	(None, 16, 32, 1024)	752024	block13_sepconv2_act[0][0]
block13_sepconv2_bn (BatchNorma	(None, 16, 32, 1024)	4096	block13_sepconv2[0][0]
conv2d_3 (Conv2D)	(None, 8, 16, 1024)	745472	add_10[0][0]
block13_pool (MaxPooling2D)	(None, 8, 16, 1024)	0	block13_sepconv2_bn[0][0]
batch_normalization_3 (BatchNor	(None, 8, 16, 1024)	4096	conv2d_3[0][0]
add_11 (Add)	(None, 8, 16, 1024)	0	block13_pool[0][0] batch_normalization_3[0][0]
block14_sepconv1 (SeparableConv	(None, 8, 16, 1536)	1582080	add_11[0][0]
block14_sepconv1_bn (BatchNorma	(None, 8, 16, 1536)	6144	block14_sepconv1[0][0]
block14_sepconv1_act (Activatio	(None, 8, 16, 1536)	0	block14_sepconv1_bn[0][0]
block14_sepconv2 (SeparableConv	(None, 8, 16, 2048)	3159552	block14_sepconv1_act[0][0]
block14_sepconv2_bn (BatchNorma	(None, 8, 16, 2048)	8192	block14_sepconv2[0][0]
block14_sepconv2_act (Activatio	(None, 8, 16, 2048)	0	block14_sepconv2_bn[0][0]
global_average_pooling2d (Globa	(None, 2048)	0	block14_sepconv2_act[0][0]
dense (Dense)	(None, 1024)	2098176	global_average_pooling2d[0][0]
batch_normalization_4 (BatchNor	(None, 1024)	4096	dense[0][0]
dropout (Dropout)	(None, 1024)	0	batch_normalization_4[0][0]
dense_1 (Dense)	(None, 512)	524800	dropout[0][0]
batch_normalization_5 (BatchNor	(None, 512)	2048	dense_1[0][0]
dropout_1 (Dropout)	(None, 512)	0	batch_normalization_5[0][0]
dense_2 (Dense)	(None, 64)	32832	dropout_1[0][0]
batch_normalization_6 (BatchNor	(None, 64)	256	dense_2[0][0]

```
dropout_2 (Dropout)             (None, 64)           0           batch_normalization_6[0][0]
_________________________________________________________________________________________________
dense_3 (Dense)                 (None, 1)            65          dropout_2[0][0]
=================================================================================================
Total params: 23,523,753
Trainable params: 2,659,073
Non-trainable params: 20,864,680
_________________________________________________________________________________________________
```

In [ ]:

```
log_dir=os.path.join("logs",datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard=tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True,write_grads=True)

checkpoint_filepath='/content/drive//My Drive/Steel_Detection /binary_Xception_2.h5'
model_checkpoint_callback=tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_filepath,monitor='val_f1_score',mode='max',save_best_only=True)
#https://keras.io/api/metrics/
#https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function
model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=["acc",f1_score])
callback=[model_checkpoint_callback,tensorboard]
#https://datascience.stackexchange.com/questions/34444/what-is-the-difference-between-fit-and-fit-generator-in-keras
history=model.fit_generator(train_image_generator,validation_data=val_image_generator,epochs=20,verbose=1,callbacks=callback)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/20
283/283 [==============================] - 4773s 17s/step - loss: 0.4982 - acc: 0.7621 - f1_score: 0.7740 - val_loss: 0.3563 - val_acc: 0.
8401 - val_f1_score: 0.8538
Epoch 2/20
283/283 [==============================] - 343s 1s/step - loss: 0.4162 - acc: 0.8027 - f1_score: 0.8147 - val_loss: 0.3349 - val_acc: 0.84
23 - val_f1_score: 0.8423
Epoch 3/20
283/283 [==============================] - 342s 1s/step - loss: 0.3764 - acc: 0.8265 - f1_score: 0.8361 - val_loss: 0.3283 - val_acc: 0.85
95 - val_f1_score: 0.8629
Epoch 4/20
283/283 [==============================] - 345s 1s/step - loss: 0.3573 - acc: 0.8351 - f1_score: 0.8448 - val_loss: 0.3268 - val_acc: 0.84
72 - val_f1_score: 0.8656
Epoch 5/20
283/283 [==============================] - 344s 1s/step - loss: 0.3553 - acc: 0.8392 - f1_score: 0.8485 - val_loss: 0.2992 - val_acc: 0.87
85 - val_f1_score: 0.8814
Epoch 6/20
283/283 [==============================] - 345s 1s/step - loss: 0.3415 - acc: 0.8451 - f1_score: 0.8535 - val_loss: 0.2955 - val_acc: 0.87
72 - val_f1_score: 0.8879
Epoch 7/20
283/283 [==============================] - 343s 1s/step - loss: 0.3219 - acc: 0.8573 - f1_score: 0.8657 - val_loss: 0.2901 - val_acc: 0.87
06 - val_f1_score: 0.8809
Epoch 8/20
283/283 [==============================] - 344s 1s/step - loss: 0.3277 - acc: 0.8523 - f1_score: 0.8600 - val_loss: 0.3000 - val_acc: 0.87
41 - val_f1_score: 0.8712
Epoch 9/20
283/283 [==============================] - 344s 1s/step - loss: 0.3263 - acc: 0.8564 - f1_score: 0.8639 - val_loss: 0.2977 - val_acc: 0.87
50 - val_f1_score: 0.8739
Epoch 10/20
283/283 [==============================] - 341s 1s/step - loss: 0.3096 - acc: 0.8694 - f1_score: 0.8761 - val_loss: 0.2669 - val_acc: 0.89
62 - val_f1_score: 0.9015
Epoch 11/20
283/283 [==============================] - 343s 1s/step - loss: 0.3026 - acc: 0.8677 - f1_score: 0.8738 - val_loss: 0.2824 - val_acc: 0.88
47 - val_f1_score: 0.8922
Epoch 12/20
283/283 [==============================] - 349s 1s/step - loss: 0.3020 - acc: 0.8625 - f1_score: 0.8700 - val_loss: 0.2718 - val_acc: 0.88
47 - val_f1_score: 0.8884
Epoch 13/20
283/283 [==============================] - 362s 1s/step - loss: 0.3046 - acc: 0.8644 - f1_score: 0.8709 - val_loss: 0.2778 - val_acc: 0.88
34 - val_f1_score: 0.8874
Epoch 14/20
283/283 [==============================] - 366s 1s/step - loss: 0.2868 - acc: 0.8794 - f1_score: 0.8848 - val_loss: 0.3023 - val_acc: 0.86
79 - val_f1_score: 0.8694
```

```
Epoch 15/20
283/283 [==============================] - 369s 1s/step - loss: 0.2880 - acc: 0.8741 - f1_score: 0.8803 - val_loss: 0.2649 - val_acc: 0.89
40 - val_f1_score: 0.8990
Epoch 16/20
283/283 [==============================] - 356s 1s/step - loss: 0.2910 - acc: 0.8738 - f1_score: 0.8792 - val_loss: 0.2654 - val_acc: 0.88
91 - val_f1_score: 0.8933
Epoch 17/20
283/283 [==============================] - 365s 1s/step - loss: 0.2843 - acc: 0.8780 - f1_score: 0.8837 - val_loss: 0.2615 - val_acc: 0.88
38 - val_f1_score: 0.8862
Epoch 18/20
283/283 [==============================] - 365s 1s/step - loss: 0.2805 - acc: 0.8802 - f1_score: 0.8856 - val_loss: 0.2603 - val_acc: 0.88
38 - val_f1_score: 0.8949
Epoch 19/20
283/283 [==============================] - 356s 1s/step - loss: 0.2939 - acc: 0.8731 - f1_score: 0.8798 - val_loss: 0.2678 - val_acc: 0.88
69 - val_f1_score: 0.8880
Epoch 20/20
283/283 [==============================] - 358s 1s/step - loss: 0.2741 - acc: 0.8831 - f1_score: 0.8886 - val_loss: 0.2369 - val_acc: 0.90
37 - val_f1_score: 0.9102
```

In [ ]:
```
%tensorboard --logdir logs
```

In [ ]:
```
model=load_model('/content/drive//My Drive/Steel_Detection /binary_Xception_2.h5',custom_objects={
'f1_score':f1_score})
```

In [ ]:
```
train_image_generator=val_datagen.flow_from_dataframe(dataframe=x_train.astype(str),
                                                      directory=train_folder_path,
                                                      x_col="image_id",
                                                      y_col="defect",
                                                      batch_size=32,
                                                      shuffle=False,
                                                      class_mode="binary",
                                                      target_size=(256,512))


val_image_generator=val_datagen.flow_from_dataframe(dataframe=x_val.astype(str),
                                                      directory=train_folder_path,
                                                      x_col="image_id",
                                                      y_col="defect",
                                                      batch_size=32,
                                                      shuffle=False,
                                                      class_mode="binary",
                                                      target_size=(256,512))

test_image_generator=val_datagen.flow_from_dataframe(dataframe=x_test.astype(str),
                                                      directory=train_folder_path,
                                                      x_col="image_id",
                                                      y_col="defect",
                                                      batch_size=32,
                                                      shuffle=False,
                                                      class_mode="binary",
                                                      target_size=(256,512))
```

```
Found 9048 validated image filenames belonging to 2 classes.
Found 2263 validated image filenames belonging to 2 classes.
Found 1257 validated image filenames belonging to 2 classes.
```

In [ ]:
```
print('Training Dataset:\n')
print(model.evaluate(train_image_generator,verbose=1))
print("="*100)
print('\nValidation Dataset:\n')
print(model.evaluate(val_image_generator,verbose=1))
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_image_generator,verbose=1))
```

```
Training Dataset:

283/283 [==============================] - 4129s 15s/step - loss: 0.2254 - acc: 0.9016 - f1_score: 0.9065
[0.22544366121292114, 0.9016121625900269, 0.9064911007881165]
====================================================================================================

Validation Dataset:

71/71 [==============================] - 1036s 15s/step - loss: 0.2369 - acc: 0.9037 - f1_score: 0.9095
[0.23688536882400513, 0.9037102460861206, 0.9095268845558167]
====================================================================================================

Test Dataset:

40/40 [==============================] - 571s 15s/step - loss: 0.2444 - acc: 0.8959 - f1_score: 0.9010
[0.24440200626850128, 0.8958664536476135, 0.9009650945663452]
```

- One can observe the value of loss and metrics are similar for train,validation and test datasets thus the model is not over-fitting. f1_score of all datasets is above 0.90. Overall model is having good performance on train, validation and test datasets.

In [ ]: