Baseline Modelling

Here we are using U-Net architecture as an baseline model with backbone as Resnet50.

```
import warnings
warnings.filterwarnings("ignore")
```

```
#installing the segmentation module
!pip install git+https://github.com/qubvel/segmentation_models
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-qekmeb4b
  Running command git clone -q https://github.com/qubvel/segmentation_models /tmp/pip-req-build-qekmeb4b
  Running command git submodule update --init --recursive -q
Collecting keras_applications<=1.0.8,>=1.0.7
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 3.4 MB/s
Collecting image-classifiers==1.0.0
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting efficientnet==1.0.0
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.8/dist-packages (from efficientnet==1.0.0->segmentation-models==1.
Requirement already satisfied: h5py in /usr/local/lib/python3.8/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentation-mode
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.8/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentat
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0.0->segment
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0.0->se
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0.0->segm
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0.0->segmentat
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0.0->segmenta
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet=
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-image->efficientnet==1.0
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-im
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib!=3.0.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->eff
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib!=3.0.0,>=2.0.0-
Building wheels for collected packages: segmentation-models
  Building wheel for segmentation-models (setup.py) ... done
  Created wheel for segmentation-models: filename=segmentation_models-1.0.1-py3-none-any.whl size=33809 sha256=965f1fc1ba29ad1cf1b5af90b
  Stored in directory: /tmp/pip-ephem-wheel-cache-624jdxzq/wheels/91/c4/cb/a53fedf4b956b22b486a1f135859dacfd3809d410e34e7906c
Successfully built segmentation-models
Installing collected packages: keras-applications, image-classifiers, efficientnet, segmentation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-1.0.8 segmentation-models-1.0.1
```

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
```

```
--2022-12-22 07:47:06--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/14241/862020/bundle/archive.zip?GoogleAccessI
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.127.128, 142.250.153.128, 142.250.145.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.127.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1684204253 (1.6G) [application/zip]
Saving to: ‘severstal-steel-defect-detection.zip’

severstal-steel-def 100%[===================>]   1.57G  40.5MB/s    in 41s

2022-12-22 07:47:47 (39.3 MB/s) - ‘severstal-steel-defect-detection.zip’ saved [1684204253/1684204253]
```

```
#unzipping the data
!unzip '/content/severstal-steel-defect-detection.zip'
```

```
Streaming output truncated to the last 5000 lines.
  inflating: train_images/99f75320d.jpg
  inflating: train_images/99f9d2375.jpg
  inflating: train_images/99fd3c6f5.jpg
  inflating: train_images/9a016fe15.jpg
  inflating: train_images/9a064450d.jpg
  inflating: train_images/9a08c2783.jpg
  inflating: train_images/9a18e4457.jpg
  inflating: train_images/9a1f7c238.jpg
  inflating: train_images/9a2523ce9.jpg
  inflating: train_images/9a3e774ff.jpg
  inflating: train_images/9a5e9e77c.jpg
  inflating: train_images/9a5f7a855.jpg
  inflating: train_images/9a62177bb.jpg
```

```
  inflating: train_images/9a67575cf.jpg
  inflating: train_images/9a6ac5406.jpg
  inflating: train_images/9a70057c0.jpg
  inflating: train_images/9a72fd89e.jpg
  inflating: train_images/9a75974ba.jpg
  inflating: train_images/9a762b892.jpg
  inflating: train_images/9a7b2f3af.jpg
  inflating: train_images/9a7b427b4.jpg
  inflating: train_images/9a81a8056.jpg
  inflating: train_images/9a83c23d1.jpg
  inflating: train_images/9a8475c90.jpg
  inflating: train_images/9a8c769b4.jpg
  inflating: train_images/9a8f98a4b.jpg
  inflating: train_images/9aa301f3e.jpg
  inflating: train_images/9aa437d47.jpg
  inflating: train_images/9aa44fa54.jpg
  inflating: train_images/9aa721852.jpg
  inflating: train_images/9aad5d03d.jpg
  inflating: train_images/9ab015391.jpg
  inflating: train_images/9ab372c8c.jpg
  inflating: train_images/9ab57a2fa.jpg
  inflating: train_images/9aca6db08.jpg
  inflating: train_images/9ad070cd0.jpg
  inflating: train_images/9ad1ad629.jpg
  inflating: train_images/9ad20b22f.jpg
  inflating: train_images/9ad36a571.jpg
  inflating: train_images/9ad4e14ae.jpg
  inflating: train_images/9adebade1.jpg
  inflating: train_images/9ae440fd0.jpg
  inflating: train_images/9aec868be.jpg
  inflating: train_images/9af9dc45b.jpg
  inflating: train_images/9afe065eb.jpg
  inflating: train_images/9b09b2a38.jpg
  inflating: train_images/9b1abd245.jpg
  inflating: train_images/9b2298962.jpg
  inflating: train_images/9b2801e98.jpg
  inflating: train_images/9b296e015.jpg
  inflating: train_images/9b2ed195e.jpg
  inflating: train_images/9b3ad1da3.jpg
  inflating: train_images/9b3d6aa5f.jpg
  inflating: train_images/9b3ef82c7.jpg
  inflating: train_images/9b40d78e0.jpg
  inflating: train_images/9b4b832e8.jpg
```

Importing Libraries

```
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 keras.utils import np_utils
from keras.utils import plot_model
from PIL import Image
import tensorflow as tf
import keras
from keras import backend as K
from keras.models import Model,load_model
from keras.regularizers import l2
import datetime
sm.set_framework('tf.keras')
sm.framework()
import segmentation_models as sm
from segmentation_models import get_preprocessing
from segmentation_models.metrics import iou_score
import imgaug.augmenters as iaa
```

```
sm.framework()
```

```
'tf.keras'
```

```
train_images_path = '/content/train_images'
train_df = pd.read_csv('/content/train.csv')
```

```
Image_id=[]
label=[]
train_folder_path='/content/train_images'
for i in os.listdir(train_folder_path): #https://www.geeksforgeeks.org/python-os-listdir-method/
    for j in range(1,5):
        Image_id.append(i)
        label.append(j)

x={'ImageId':Image_id,'ClassId':label}  #https://www.geeksforgeeks.org/creating-a-pandas-dataframe/
train_img=pd.DataFrame(x)
train_img.head(10)
```

	ImageId	ClassId
0	c62b766cc.jpg	1
1	c62b766cc.jpg	2
2	c62b766cc.jpg	3
3	c62b766cc.jpg	4
4	15796b4d5.jpg	1
5	15796b4d5.jpg	2
6	15796b4d5.jpg	3
7	15796b4d5.jpg	4
8	24aa8d834.jpg	1
9	24aa8d834.jpg	2

```
#https://stackoverflow.com/questions/53645882/pandas-merging-10
df=pd.merge(train_img,train_df,how='outer',on=['ImageId','ClassId'])
df.fillna('',inplace=True)
df.head()
```

	ImageId	ClassId	EncodedPixels
0	c62b766cc.jpg	1	
1	c62b766cc.jpg	2	
2	c62b766cc.jpg	3	
3	c62b766cc.jpg	4	
4	15796b4d5.jpg	1	

```
#https://www.analyticsvidhya.com/blog/2020/03/pivot-table-pandas-python/
train=pd.pivot_table(df,values='EncodedPixels',index='ImageId',columns='ClassId',aggfunc=np.sum).astype(str)
train=train.reset_index()
train.columns=['image_id','rle_1','rle_2','rle_3','rle_4']
train.head()
```

	image_id	rle_1	rle_2	rle_3	rle_4
0	0002cc93b.jpg	29102 12 29346 24 29602 24 29858 24 30114 24 3...			
1	00031f466.jpg				
2	000418bfc.jpg				
3	000789191.jpg				
4	0007a71bf.jpg			18661 28 18863 82 19091 110	

```
defect=[]
stratify=[]
for i in range(len(train)):
```

```
  if (train['rle_1'][i] != '' or train['rle_2'][i] != '' or train['rle_3'][i] != '' or train['rle_4'][i] != ''):
    defect.append(1)
  else:
    defect.append(0)

  if train['rle_1'][i] != '':
    stratify.append(1)
  elif train['rle_2'][i] != '':
    stratify.append(2)
  elif train['rle_3'][i] != '':
    stratify.append(3)
  elif train['rle_4'][i] != '':
    stratify.append(4)
  else:
    stratify.append(0)
train['defect']=defect
train['stratify']=stratify


defect_1,defect_2,defect_3,defect_4=[],[],[],[]
for i in range(len(train)):
  if train['rle_1'][i] != '':
    defect_1.append(1)
  else:
    defect_1.append(0)
  if train['rle_2'][i] != '':
    defect_2.append(1)
  else:
    defect_2.append(0)
  if train['rle_3'][i] != '':
    defect_3.append(1)
  else:
    defect_3.append(0)
  if train['rle_4'][i] != '':
    defect_4.append(1)
  else:
    defect_4.append(0)
train['defect_1']=defect_1
train['defect_2']=defect_2
train['defect_3']=defect_3
train['defect_4']=defect_4
train['total_defects']=train['defect_1']+ train['defect_2']+ train['defect_3']+ train['defect_4']
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

18661 28 18863

```
with open('/content/train.pkl','rb') as f:
    train=pickle.load(f)


train.head()
```

	image_id	rle_1	rle_2	rle_3	rle_4	defect	stratify	defect_1	defect_2	defec
0	0002cc93b.jpg	29102 12 29346 24 29602 24 29858 24 30114 24 3...				1	1	1	0	

Train-Test Split

```
#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)
print("x_train {}".format(x_train.shape),"x_test {}".format(x_test.shape))
print("="*100)
```

```
x_train (11311, 12) x_test (1257, 12)
====================================================================================================
```

Defect-1 train and test data

```
train_1=x_train[x_train['defect_1']==1][['image_id','rle_1']].rename(columns={'image_id':'image_id','rle_1':'rle'})
train_2=x_train[x_train['defect_2']==1][['image_id','rle_2']].rename(columns={'image_id':'image_id','rle_2':'rle'})
train_3=x_train[x_train['defect_3']==1][['image_id','rle_3']].rename(columns={'image_id':'image_id','rle_3':'rle'})
train_4=x_train[x_train['defect_4']==1][['image_id','rle_4']].rename(columns={'image_id':'image_id','rle_4':'rle'})
print("train_1 {}".format(train_1.shape)," train_2 {}".format(train_2.shape)," train_3 {}".format(train_3.shape)," train_4 {}".format(train_4
print("="*100)
```

```
train_1 (807, 2)  train_2 (224, 2)  train_3 (4638, 2)  train_4 (723, 2)
====================================================================================================
```

```
train_4.to_pickle("train_4.pkl")
```

```
test_1=x_test[x_test['defect_1']==1][['image_id','rle_1']].rename(columns={'image_id':'image_id','rle_1':'rle'})
test_2=x_test[x_test['defect_2']==1][['image_id','rle_2']].rename(columns={'image_id':'image_id','rle_2':'rle'})
test_3=x_test[x_test['defect_3']==1][['image_id','rle_3']].rename(columns={'image_id':'image_id','rle_3':'rle'})
test_4=x_test[x_test['defect_4']==1][['image_id','rle_4']].rename(columns={'image_id':'image_id','rle_4':'rle'})
print("test_1 {}".format(test_1.shape)," test_2 {}".format(test_2.shape)," test_3 {}".format(test_3.shape)," test_4 {}".format(test_4.shape))
```

```
test_1 (90, 2)  test_2 (23, 2)  test_3 (512, 2)  test_4 (78, 2)
```

```
with open('/content/train_4.pkl','rb') as f:
    train_4=pickle.load(f)
```

```
#https://www.kaggle.com/paulorzp/rle-functions-run-lenght-encode-decode
def rle_to_mask(rle):
# CONVERT RLE TO MASK
   if (pd.isnull(rle))|(rle==''')|(rle=='-1'):
       return np.zeros((256,800),dtype=np.uint8) #If the EncodedPixels string is empty an empty mask is returned
   height=256
   width=1600
   mask=np.zeros(width*height,dtype=np.uint8)
   array=np.asarray([int(x) for x in rle.split()])
   starts=array[0::2]-1
   lengths=array[1::2]
   for index,start in enumerate(starts):
      mask[int(start):int(start+lengths[index])]=1
   return mask.reshape((height,width),order='F')[::,::2]
```

Train and Test generators

```
"""
Implementing custom data generator
#https://towardsdatascience.com/implementing-custom-data-generators-in-keras-de56f013581c
#https://www.kaggle.com/cdeotte/keras-unet-with-eda
"""
class train_DataGenerator(keras.utils.Sequence):
    def __init__(self,dataframe,batch_size=1,shuffle=True,preprocess=None,info={}):
     self.batch_size = batch_size
     self.df = dataframe
     self.indices = self.df.index.tolist()
     self.preprocess = preprocess
     self.shuffle = shuffle
     self.on_epoch_end()
    def __len__(self):
     return len(self.indices) // (self.batch_size)
    def __getitem__(self, index):
     index = self.index[index * self.batch_size:(index + 1) * self.batch_size]
```

```
     batch = [self.indices[k] for k in index]
     X, y = self.__get_data(batch)
     return X, y
    def on_epoch_end(self):
     self.index = np.arange(len(self.indices))
     if self.shuffle == True:
        np.random.shuffle(self.index)
    def __get_data(self, batch):
     train_datagen = ImageDataGenerator()
#https://www.geeksforgeeks.org/python-select-random-value-from-a-list/
     X=np.empty((self.batch_size,256,800,3),dtype=np.float32) # image place-holders
     Y=np.empty((self.batch_size,256,800,1),dtype=np.float32)# 1 mask place-holders
     for i,id in enumerate(batch):
      X[i,] = Image.open('/content/train_images/' + str(self.df['image_id'].loc[id])).resize((800,256))
      Y[i,:,:,0]=rle_to_mask(self.df['rle'].loc[id])
      t=random.choice([0,10,20,30,40])
      z=random.choice([0.8,1])
      flip=random.choice(['True','False'])
      param={'tx':t,'ty':t,'zx':z,'zy':z,}
      for i,e in enumerate(X):
          X[i] = train_datagen.apply_transform(e,transform_parameters=param)
      for i,f in enumerate(Y):
          Y[i] = train_datagen.apply_transform(f,transform_parameters=param)
      if self.preprocess!=None: X = self.preprocess(X)
      return X,Y

# Implementing custom data generator
#https://towardsdatascience.com/implementing-custom-data-generators-in-keras-de56f013581c
class test_DataGenerator(keras.utils.Sequence):
  def __init__(self,dataframe,batch_size=1,shuffle=False,preprocess=None,info={}):
   self.batch_size = batch_size
   self.df = dataframe
   self.indices = self.df.index.tolist()
   self.preprocess = preprocess
   self.shuffle = shuffle
   self.on_epoch_end()
  def __len__(self):
   return len(self.indices) // (self.batch_size)
  def __getitem__(self, index):
   index = self.index[index * self.batch_size:(index + 1) * self.batch_size]
   batch = [self.indices[k] for k in index]
   X, y = self.__get_data(batch)
   return X, y
  def on_epoch_end(self):
   self.index = np.arange(len(self.indices))
   if self.shuffle == True:
     np.random.shuffle(self.index)
  def __get_data(self, batch):
   X = np.empty((self.batch_size,256,800,3),dtype=np.float32) # image place-holders
   Y = np.empty((self.batch_size,256,800,1),dtype=np.float32)# 1 mask place-holders
   for i, id in enumerate(batch):
     X[i,] = Image.open('/content/train_images/' + str(self.df['image_id'].loc[id])).resize((800,256))
     Y[i,:,:,0]=rle_to_mask(self.df['rle'].loc[id])
    # preprocess input
   if self.preprocess!=None: X = self.preprocess(X)
   return X,Y


train_4[1:5]
```

	image_id	rle
11840	f0dbecd3d.jpg	169742 2 169997 5 170252 9 170507 12 170762 15...
9335	bd4957b80.jpg	129032 1 129288 2 129544 2 129800 3 130056 3 1...
3640	49da138ee.jpg	78079 2 78333 4 78587 6 78841 8 79095 10 79349...
10729	d96c0e264.jpg	326648 1 326904 3 327159 5 327415 6 327670 9 3...

U-net with backbone as Resnet50

```
preprocess=get_preprocessing('mobilenet')
model=sm.Unet('mobilenet',classes=1,activation='sigmoid',encoder_weights='imagenet',encoder_freeze = True,decoder_filters=(48,32,16,8,4))
model._name="Segmentation_Model"
model.summary()
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.6/mobilenet_1_0_224_tf_no_top.h5
17225924/17225924 [==============================] - 2s 0us/step
Model: "Segmentation_Model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, None, None,  0           []
                                 3)]

 conv1_pad (ZeroPadding2D)      (None, None, None,   0           ['input_1[0][0]']
                                3)

 conv1 (Conv2D)                 (None, None, None,   864         ['conv1_pad[0][0]']
                                32)

 conv1_bn (BatchNormalization)  (None, None, None,   128         ['conv1[0][0]']
                                32)

 conv1_relu (ReLU)              (None, None, None,   0           ['conv1_bn[0][0]']
                                32)

 conv_dw_1 (DepthwiseConv2D)    (None, None, None,   288         ['conv1_relu[0][0]']
                                32)

 conv_dw_1_bn (BatchNormalizati  (None, None, None,   128        ['conv_dw_1[0][0]']
 on)                            32)

 conv_dw_1_relu (ReLU)          (None, None, None,   0           ['conv_dw_1_bn[0][0]']
                                32)

 conv_pw_1 (Conv2D)             (None, None, None,   2048        ['conv_dw_1_relu[0][0]']
                                64)

 conv_pw_1_bn (BatchNormalizati  (None, None, None,   256        ['conv_pw_1[0][0]']
 on)                            64)

 conv_pw_1_relu (ReLU)          (None, None, None,   0           ['conv_pw_1_bn[0][0]']
                                64)

 conv_pad_2 (ZeroPadding2D)     (None, None, None,   0           ['conv_pw_1_relu[0][0]']
                                64)

 conv_dw_2 (DepthwiseConv2D)    (None, None, None,   576         ['conv_pad_2[0][0]']
                                64)

 conv_dw_2_bn (BatchNormalizati  (None, None, None,   256        ['conv_dw_2[0][0]']
 on)                            64)

 conv_dw_2_relu (ReLU)          (None, None, None,   0           ['conv_dw_2_bn[0][0]']
                                64)

 conv_pw_2 (Conv2D)             (None, None, None,   8192        ['conv_dw_2_relu[0][0]']
                                128)

 conv_pw_2_bn (BatchNormalizati  (None, None, None,   512        ['conv_pw_2[0][0]']
 on)                            128)

 conv_pw_2_relu (ReLU)          (None, None, None,   0           ['conv_pw_2_bn[0][0]']
```

```
train_batch=train_DataGenerator(train_4,shuffle=True,preprocess=preprocess)
valid_batch=test_DataGenerator(test_4,preprocess=preprocess)
#callbacks
log_dir = os.path.join("logs",'fits', 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)
loss = sm.losses.bce_dice_loss
#defining the optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
#compiling the model
model.compile(optimizer, loss, metrics = [iou_score])
#defining the callbacks
cb = [tf.keras.callbacks.TensorBoard(log_dir,histogram_freq = 1,write_graph = True),

      tf.keras.callbacks.ModelCheckpoint(monitor = 'val_iou_score',
                                         filepath = 'model_unet.h5',
                                         save_best_only=True),
      ]
history=model.fit_generator(train_batch,validation_data=valid_batch,epochs=5,verbose=1,callbacks=cb)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/5
723/723 [==============================] - 65s 73ms/step - loss: 0.7209 - iou_score: 0.3441 - val_loss: 0.5776 - val_iou_score: 0.4393
Epoch 2/5
723/723 [==============================] - 54s 74ms/step - loss: 0.5840 - iou_score: 0.4392 - val_loss: 0.5035 - val_iou_score: 0.5086
Epoch 3/5
723/723 [==============================] - 54s 74ms/step - loss: 0.5325 - iou_score: 0.4803 - val_loss: 0.4944 - val_iou_score: 0.4898
Epoch 4/5
723/723 [==============================] - 55s 76ms/step - loss: 0.4955 - iou_score: 0.5038 - val_loss: 0.5171 - val_iou_score: 0.4954
Epoch 5/5
723/723 [==============================] - 56s 77ms/step - loss: 0.4689 - iou_score: 0.5259 - val_loss: 0.4997 - val_iou_score: 0.5197
```

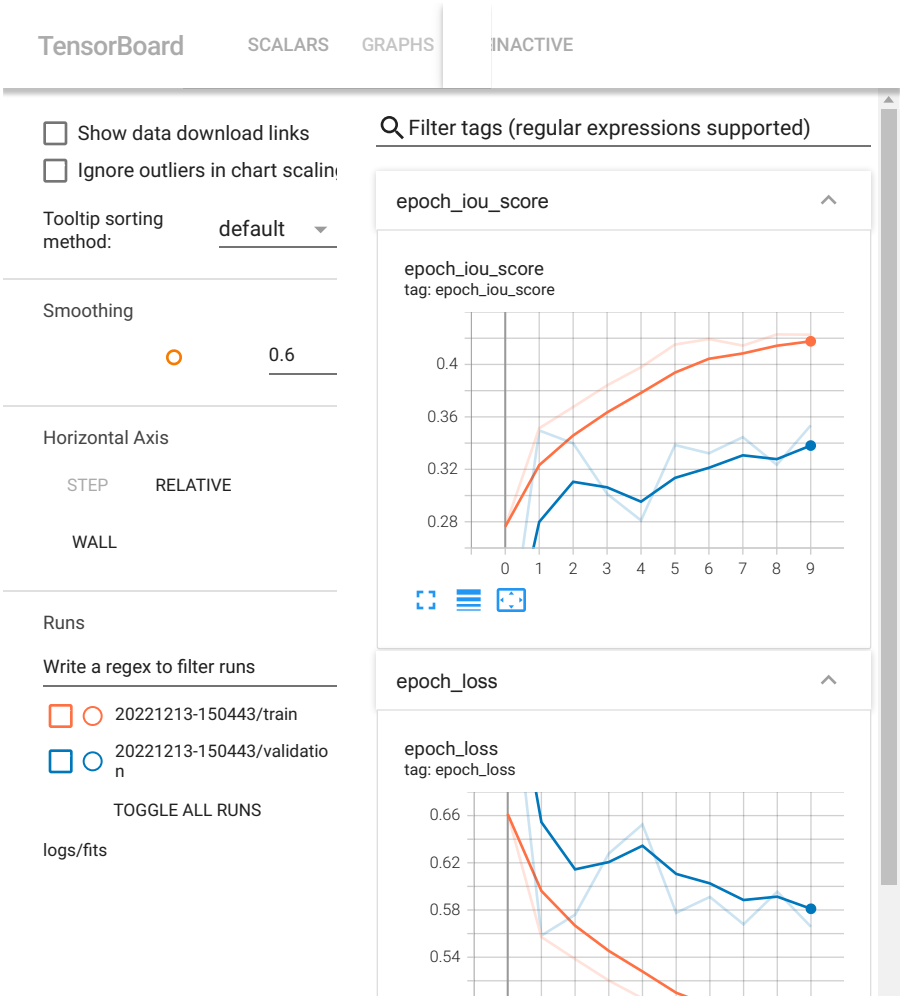
```
UNet = tf.keras.models.load_model('/content/model_unet.h5',custom_objects={'binary_crossentropy_plus_dice_loss':sm.losses.bce_dice_loss,'iou_
tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(UNet)
tflite_UNet = tf_lite_converter.convert()
open('UNet.tflite', "wb").write(tflite_UNet )
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op,
17035160
```

```
train_batch=train_DataGenerator(train_1,shuffle=True,preprocess=preprocess)
valid_batch=test_DataGenerator(test_1,preprocess=preprocess)
#callbacks
log_dir = os.path.join("logs",'fits', 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)
loss = sm.losses.bce_dice_loss
#defining the optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
#compiling the model
model.compile(optimizer, loss, metrics = [iou_score])
#defining the callbacks
cb = [tf.keras.callbacks.TensorBoard(log_dir,histogram_freq = 1,write_graph = True),

      tf.keras.callbacks.ModelCheckpoint(monitor = 'val_iou_score',
                                         filepath = 'model_unet1.h5',
                                         save_best_only = True),
      ]
history=model.fit_generator(train_batch,validation_data=valid_batch,epochs=5,verbose=1,callbacks=cb)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/5
807/807 [==============================] - 103s 105ms/step - loss: 0.6723 - iou_score: 0.2754 - val_loss: 0.5458 - val_iou_score: 0.3615
Epoch 2/5
807/807 [==============================] - 94s 116ms/step - loss: 0.5567 - iou_score: 0.3540 - val_loss: 0.5589 - val_iou_score: 0.3445
Epoch 3/5
807/807 [==============================] - 107s 133ms/step - loss: 0.5309 - iou_score: 0.3768 - val_loss: 0.6455 - val_iou_score: 0.2906
Epoch 4/5
807/807 [==============================] - 105s 131ms/step - loss: 0.5216 - iou_score: 0.3844 - val_loss: 0.6965 - val_iou_score: 0.2509
Epoch 5/5
807/807 [==============================] - 107s 133ms/step - loss: 0.5064 - iou_score: 0.3972 - val_loss: 0.5781 - val_iou_score: 0.3383
```

```
%tensorboard --logdir logs/fits
```




```
print('Training Dataset:\n')
print(model.evaluate(test_DataGenerator(train_1,preprocess=preprocess),verbose=1))
print("="*100)
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_DataGenerator(test_1,preprocess=preprocess),verbose=1))
```

```
Training Dataset:

807/807 [==============================] - 33s 41ms/step - loss: 0.4358 - iou_score: 0.4614
[0.43580207228660583, 0.4614080488681793]
====================================================================================================
====================================================================================================

Test Dataset:

90/90 [==============================] - 4s 41ms/step - loss: 0.4860 - iou_score: 0.4153
[0.4860096871852875, 0.41534867882728577]
```

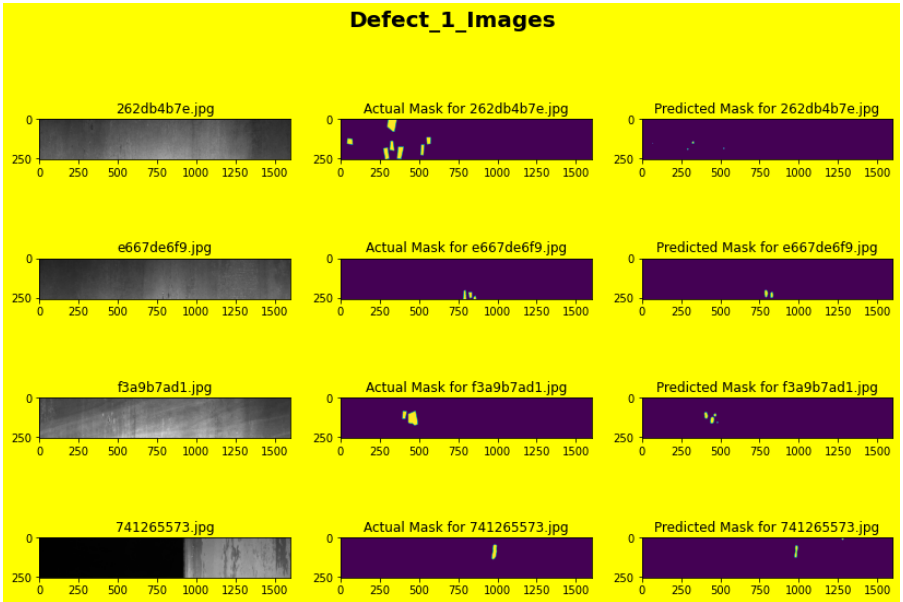
```
print('Training Dataset:\n')
print(model.evaluate(test_DataGenerator(train_3,preprocess=preprocess),verbose=1))
print("="*100)
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_DataGenerator(test_3,preprocess=preprocess),verbose=1))
```

```
Training Dataset:

4638/4638 [==============================] - 81s 18ms/step - loss: 0.6348 - iou_score: 0.3779
[0.6347724795341492, 0.3779484033584595]
====================================================================================================
====================================================================================================

Test Dataset:

512/512 [==============================] - 9s 17ms/step - loss: 0.6488 - iou_score: 0.3783
[0.6487862467765808, 0.37832674384117126]
```

```
def rle2mask(rle):
# CONVERT RLE TO MASK
   if (pd.isnull(rle))|(rle==''')|(rle=='-1'):
      return np.zeros((256,1600) ,dtype=np.uint8)
   height= 256
   width = 1600
   mask= np.zeros( width*height ,dtype=np.uint8)
   array = np.asarray([int(x) for x in rle.split()])
   starts = array[0::2]-1
   lengths = array[1::2]
   for index, start in enumerate(starts):
      mask[int(start):int(start+lengths[index])] = 1
   return mask.reshape( (height,width), order='F' )


def plot_mask(rle_defect,k,pred):
   train_folder_path='/content/train_images/'
   # Create figure and axes
   fig,ax=plt.subplots(4,3,figsize=(14,9))
   fig.suptitle('Defect_'+str(k)+'_Images',fontsize=20,fontweight='bold')
   for i in range(4):
     image_id=rle_defect[i][0]
     rle=rle_defect[i][1]
     im=Image.open(train_folder_path+str(image_id))
     ax[i,0].imshow(im)
     ax[i,0].set_title(image_id)
     mask=rle2mask(rle)
     ax[i,1].imshow(mask)
     ax[i,1].set_title("Actual Mask for "+str(image_id))
     c1=Image.fromarray(pred[i][:,:,0])
     ax[i,2].imshow(np.array(c1.resize((1600,256)))>0.5)
     ax[i,2].set_title("Predicted Mask for "+str(image_id))
   fig.set_facecolor("yellow")
   plt.show()


train_preds=model.predict_generator(test_DataGenerator(train_1[9:13],preprocess=preprocess),verbose=1)

     4/4 [==============================] - 1s 55ms/step
```

Plotting Defect -1 images

```
plot_mask(train_1[9:13].values,1,train_preds)
```



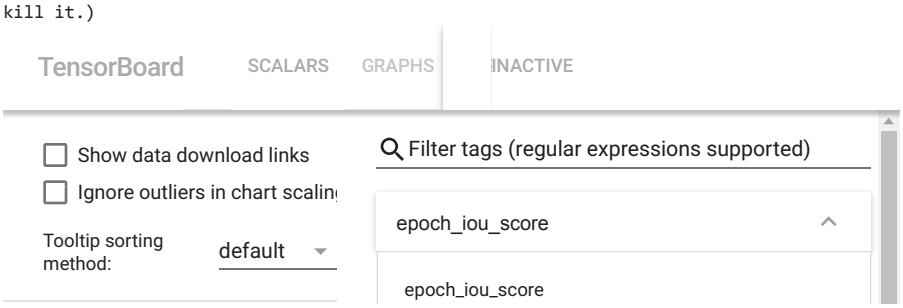
Defect-2

```
train_batch=train_DataGenerator(train_2,shuffle=True,preprocess=preprocess)
valid_batch=test_DataGenerator(test_2,preprocess=preprocess)
#callbacks
log_dir = os.path.join("logs",'fits', 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)
#defining the loss
loss = sm.losses.bce_dice_loss
#defining the optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
#compiling the model
model.compile(optimizer, loss, metrics = [iou_score])
#defining the callbacks
cb = [tf.keras.callbacks.TensorBoard(log_dir,histogram_freq = 1,write_graph = True),

      tf.keras.callbacks.ModelCheckpoint(monitor = 'val_iou_score',
                                         filepath = 'model_unet2.h5',
                                         save_best_only = True),
      ]
history=model.fit_generator(train_batch,validation_data=valid_batch,epochs=10,verbose=1,callbacks=cb)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/10
224/224 [==============================] - 48s 135ms/step - loss: 0.8987 - iou_score: 0.1047 - val_loss: 0.7968 - val_iou_score: 0.1607
Epoch 2/10
224/224 [==============================] - 33s 146ms/step - loss: 0.6201 - iou_score: 0.2928 - val_loss: 0.8647 - val_iou_score: 0.1243
Epoch 3/10
224/224 [==============================] - 35s 155ms/step - loss: 0.5835 - iou_score: 0.3244 - val_loss: 0.9691 - val_iou_score: 0.0715
Epoch 4/10
224/224 [==============================] - 34s 153ms/step - loss: 0.5656 - iou_score: 0.3421 - val_loss: 0.7151 - val_iou_score: 0.2409
Epoch 5/10
224/224 [==============================] - 34s 152ms/step - loss: 0.5492 - iou_score: 0.3554 - val_loss: 0.6017 - val_iou_score: 0.3297
Epoch 6/10
224/224 [==============================] - 34s 150ms/step - loss: 0.5073 - iou_score: 0.3903 - val_loss: 0.6227 - val_iou_score: 0.3205
Epoch 7/10
224/224 [==============================] - 33s 146ms/step - loss: 0.5144 - iou_score: 0.3883 - val_loss: 0.6965 - val_iou_score: 0.2675
Epoch 8/10
224/224 [==============================] - 34s 151ms/step - loss: 0.5053 - iou_score: 0.3974 - val_loss: 0.6048 - val_iou_score: 0.3335
Epoch 9/10
224/224 [==============================] - 33s 147ms/step - loss: 0.5050 - iou_score: 0.3986 - val_loss: 0.7143 - val_iou_score: 0.2440
Epoch 10/10
224/224 [==============================] - 35s 155ms/step - loss: 0.4970 - iou_score: 0.4052 - val_loss: 0.5641 - val_iou_score: 0.3570
```

```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 1916), started 0:29:25 ago. (Use '!kill 1916' to
kill it.)
```




```
model=load_model('/content/model_unet2.h5',custom_objects={'binary_crossentropy_plus_dice_loss':sm.losses.bce_dice_loss,'iou_score':iou_score
```

```
print('Training Dataset:\n')
print(model.evaluate(test_DataGenerator(train_2,preprocess=preprocess),verbose=1))
print("="*100)
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_DataGenerator(test_2,preprocess=preprocess),verbose=1))
```

```
Training Dataset:

224/224 [==============================] - 10s 38ms/step - loss: 0.8709 - iou_score: 0.1342
[0.8709443211555481, 0.13424713909626007]
================================================================================================================
================================================================================================================

Test Dataset:

23/23 [==============================] - 1s 44ms/step - loss: 0.9691 - iou_score: 0.0715
[0.9690846800804138, 0.07150447368621826]
```

TOGGLE ALL RUNS

```
train_preds=model.predict_generator(test_DataGenerator(train_2[10:14],preprocess=preprocess),verbose=1)
```

```
4/4 [==============================] - 0s 47ms/step
```

```
plot_mask(train_2[10:14].values,1,train_preds)
```

Defect-3

```
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
train_batch=train_DataGenerator(train_3,shuffle=True,preprocess=preprocess)
valid_batch=test_DataGenerator(test_3,preprocess=preprocess)
#callbacks
log_dir = os.path.join("logs",'fits', 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)
#https://keras.io/api/metrics/
#https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function
#defining the loss
loss = sm.losses.bce_dice_loss
#defining the optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
#compiling the model
model.compile(optimizer, loss, metrics = [iou_score])
#defining the callbacks
cb = [tf.keras.callbacks.TensorBoard(log_dir,histogram_freq = 1,write_graph = True),

      tf.keras.callbacks.ModelCheckpoint(monitor = 'val_iou_score',
                                         filepath = 'model_unet3.h5',
                                         save_best_only = True), ]
history=model.fit_generator(train_batch,validation_data=valid_batch,epochs=5,verbose=1,callbacks=cb)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
Epoch 1/5
4638/4638 [==============================] - 553s 118ms/step - loss: 0.6458 - iou_score: 0.3869 - val_loss: 0.6562 - val_iou_score: 0.37
Epoch 2/5
4638/4638 [==============================] - 545s 118ms/step - loss: 0.5481 - iou_score: 0.4502 - val_loss: 0.8392 - val_iou_score: 0.24
Epoch 3/5
4638/4638 [==============================] - 540s 116ms/step - loss: 0.5292 - iou_score: 0.4637 - val_loss: 0.6666 - val_iou_score: 0.38
Epoch 4/5
4638/4638 [==============================] - 546s 118ms/step - loss: 0.5069 - iou_score: 0.4785 - val_loss: 0.6762 - val_iou_score: 0.36
Epoch 5/5
4638/4638 [==============================] - 547s 118ms/step - loss: 0.4937 - iou_score: 0.4875 - val_loss: 0.7720 - val_iou_score: 0.28
```

```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 522), started 0:55:05 ago. (Use '!kill 522' to kill it.)
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS TIME SERIES INACTIVE

```
model=load_model('/content/model_unet3.h5',custom_objects={'binary_crossentropy_plus_dice_loss':sm.losses.bce_dice_loss,'iou_score':iou_score
```

Show data download links Filter tags (regular expressions supported)

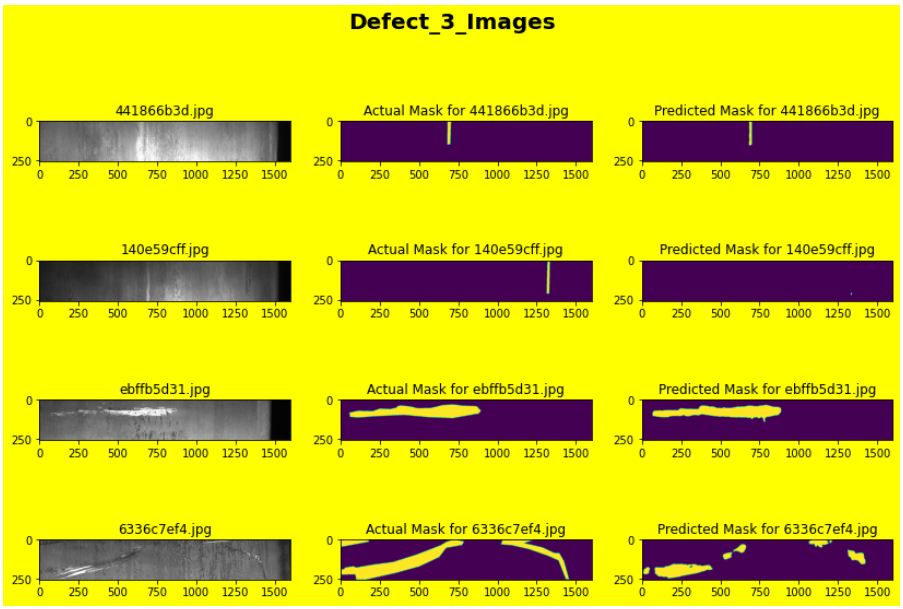
```
print('Training Dataset:\n')
print(model.evaluate(test_DataGenerator(train_3,preprocess=preprocess),verbose=1))
print("="*100)
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_DataGenerator(test_3,preprocess=preprocess),verbose=1))
```

```
Training Dataset:

4638/4638 [==============================] - 182s 39ms/step - loss: 0.7250 - iou_score: 0.3039
[0.7250118851661682, 0.3038969933986664]
====================================================================================================
====================================================================================================

Test Dataset:

512/512 [==============================] - 21s 41ms/step - loss: 0.7368 - iou_score: 0.3034
[0.7368038892745972, 0.3033551275730133]
```

```
train_preds=model.predict_generator(test_DataGenerator(train_3[9:13],preprocess=preprocess),verbose=1)
```

```
4/4 [==============================] - 1s 48ms/step
```

```
plot_mask(train_3[9:13].values,3,train_preds)
```



Defect-4

```
train_batch=train_DataGenerator(train_4,shuffle=True,preprocess=preprocess)
valid_batch=test_DataGenerator(test_4,preprocess=preprocess)
#callbacks
log_dir = os.path.join("logs",'fits', 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)
#https://keras.io/api/metrics/
#https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function
#defining the loss
loss = sm.losses.bce_dice_loss
#defining the optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
```

```
#compiling the model
model.compile(optimizer, loss, metrics = [iou_score])
#defining the callbacks
cb = [tf.keras.callbacks.TensorBoard(log_dir,histogram_freq = 1,write_graph = True),

      tf.keras.callbacks.ModelCheckpoint(monitor = 'val_iou_score',
                                         filepath = 'model_unet4.h5',
                                         save_best_only = True),
      ]
#https://datascience.stackexchange.com/questions/34444/what-is-the-difference-between-fit-and-fit-generator-in-keras
history=model.fit_generator(train_batch,validation_data=valid_batch,epochs=10,verbose=1,callbacks=cb)
```

```
     WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
     Epoch 1/10
     723/723 [==============================] - 97s 126ms/step - loss: 0.5635 - iou_score: 0.4506 - val_loss: 0.5990 - val_iou_score: 0.4120
     Epoch 2/10
     723/723 [==============================] - 93s 129ms/step - loss: 0.4844 - iou_score: 0.5082 - val_loss: 0.5341 - val_iou_score: 0.4561
     Epoch 3/10
     723/723 [==============================] - 92s 128ms/step - loss: 0.4471 - iou_score: 0.5373 - val_loss: 0.4867 - val_iou_score: 0.4991
     Epoch 4/10
     723/723 [==============================] - 96s 132ms/step - loss: 0.4281 - iou_score: 0.5513 - val_loss: 0.5137 - val_iou_score: 0.4692
     Epoch 5/10
     723/723 [==============================] - 96s 132ms/step - loss: 0.4223 - iou_score: 0.5599 - val_loss: 0.5236 - val_iou_score: 0.4649
     Epoch 6/10
     723/723 [==============================] - 94s 130ms/step - loss: 0.4197 - iou_score: 0.5595 - val_loss: 0.5207 - val_iou_score: 0.4574
     Epoch 7/10
     723/723 [==============================] - 93s 129ms/step - loss: 0.3955 - iou_score: 0.5801 - val_loss: 0.4976 - val_iou_score: 0.5033
     Epoch 8/10
     723/723 [==============================] - 91s 126ms/step - loss: 0.3981 - iou_score: 0.5770 - val_loss: 0.4221 - val_iou_score: 0.5594
     Epoch 9/10
     723/723 [==============================] - 96s 132ms/step - loss: 0.3955 - iou_score: 0.5809 - val_loss: 0.4316 - val_iou_score: 0.5461
     Epoch 10/10
     723/723 [==============================] - 94s 130ms/step - loss: 0.3956 - iou_score: 0.5772 - val_loss: 0.5304 - val_iou_score: 0.4588
```

```
train_preds=model.predict_generator(test_DataGenerator(train_4[9:13],preprocess=preprocess),verbose=1)
```

```
     4/4 [==============================] - 1s 58ms/step
```

```
%tensorboard --logdir logs/fits
```

TensorBoard SCALARS GRAPHS INACTIVE

```
print('Training Dataset:\n')
print(model.evaluate(test_DataGenerator(train_4,preprocess=preprocess),verbose=1))
print("="*100)
print("="*100)
print('\nTest Dataset:\n')
print(model.evaluate(test_DataGenerator(test_4,preprocess=preprocess),verbose=1))
```

```
Training Dataset:

723/723 [==============================] - 28s 39ms/step - loss: 0.5408 - iou_score: 0.4679
[0.5408163070678711, 0.46790897846221924]
====================================================================================================
====================================================================================================

Test Dataset:

78/78 [==============================] - 3s 39ms/step - loss: 0.5304 - iou_score: 0.4588
[0.530367910861969, 0.45882436633110046]
```

```
plot_mask(train_4[9:13].values,4,train_preds)
```