Problem statement:

Given an image of steel sheet find the type of defect from one of the four types of defects defined.

This project was motivated from the kaggle competition hosted by Severstal.

Data Source

Source : kaggle . Click below to view the data source .

[Click here]

Metric used.

In this competition we are trying to maximize the dice coeffieient.

In [3]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%
b&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.
2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fww
ogleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
from tqdm import tqdm_notebook
import cv2

import keras
from keras.layers.convolutional import Conv2DTranspose
from keras.layers.merge import concatenate
from keras.layers import UpSampling2D, Conv2D, Activation, Input, Dropout, MaxPooling2D
from keras import Model
from keras import backend as K
from keras.layers.core import Lambda
from PIL import Image
import warnings
warnings.filterwarnings("ignore")
```

```
Using TensorFlow backend.
```

In [0]:

```
import os
lst = os.listdir("/content/drive/My Drive/train_images")
```

In [6]:

```
import pandas as pd
train = pd.read_csv("/content/drive/My Drive/Project/train1.csv")
```

```
train = pd.read_csv('/content/drive/My Drive/Project/train.csv')
train.shape
```

Out[6]:

```
(50272, 2)
```

Structuring the dataset

In [7]:

```
# Now we will structure the data. Currently we have four entries for each image corresponding to e
ach class(1 to 4) with ite RLE.
# We will convert this to one row for each image with four columns of RLE corresponding to each cl
ass.
train['ImageId'] = train['ImageId_ClassId'].map(lambda x : x.split(".")[0]+'.jpg') # This will take
id after interval of 4 rows.
n_train = pd.DataFrame({'ImageId':train['ImageId'][0::4]}) # Creating dataframe with image names o
f images
n_train['e1'] = train['EncodedPixels'][0::4].values # Will take encoding after interval of 4.
n_train['e2'] = train['EncodedPixels'][1::4].values
n_train['e3'] = train['EncodedPixels'][2::4].values
n_train['e4'] = train['EncodedPixels'][3::4].values
n_train.reset_index(inplace=True, drop=True)
n_train.fillna('', inplace=True)
n_train.head()
```

Out[7]:

	ImageId	e1	e2	e3	e4
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 19347 110 19603 11...	

In [0]:

```
train_df = n_train.iloc[:int(0.80*len(n_train))]
test_df = n_train.iloc[int(0.80*len(n_train)):int(0.9*len(n_train))]
cv_df = n_train.iloc[int(0.9*len(n_train)):]
```

In [9]:

```
print(train_df.shape)
print(test_df.shape)
print(cv_df.shape)
```

```
(10054, 5)
(1257, 5)
(1257, 5)
```

In [0]:

```
# Function to convert run length encoding(rle) to mask.
# Mask covers the image by coloring the pixels that are to be highlighted.
import numpy as np
def rle2mask(rle):
  # If rle is empty or null
  if(len(rle)<1):
    return np.zeros((128,800) ,dtype=np.uint8)

  height = 256
  width = 1600
```

```
  width = 1600

  # Defining the length of mask. This will be 1d array and later will be reshaped to 2d.
  mask = np.zeros(height*width ).astype(np.uint8)
  # We will have an array that wil contain rle
  array = np.asarray([int(x) for x in rle.split()])
  start = array[0::2]-1 # this willl contain the start of run length
  length = array[1::2] # this will contain the length of each rle.

  #  now we will chane the value of each pixel in the rle to 1.
  for i,start in enumerate(start):
    mask[int(start):int(start+length[i])] = 1

  # now we will return the mask by first reshaping it and then rotating by 90 degrees and the vert
ically flipping it upside down.
  #return np.flipud(np.rot90(mask.reshape(width, height), k=1)) # Here k=1 means we will rotate on
ly once.
  return mask.reshape( (height,width), order='F' )[::2,::2]
```

In [0]:

```
def mask2rle(img):
    '''
    img: numpy array, 1 - mask, 0 - background
    Returns run length as string formated
    '''
    #print(img.shape)
    pixels= img.T.flatten()
    pixels = np.concatenate([[0], pixels, [0]])
    runs = np.where(pixels[1:] != pixels[:-1])[0] + 1
    runs[1::2] -= runs[::2]
    return ' '.join(str(x) for x in runs)
```

In [0]:

```
# https://www.kaggle.com/ateplyuk/pytorch-starter-u-net-resnet
# https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly
import keras
from keras.preprocessing.image import ImageDataGenerator



class DataGenerator(keras.utils.Sequence):
    def __init__(self, df, batch_size = 16, subset="train", shuffle=False,
                 preprocess=None, info={}):
        super().__init__()
        self.df = df
        self.shuffle = shuffle
        self.subset = subset
        self.batch_size = batch_size
        self.preprocess = preprocess
        self.info = info

        if self.subset == "train":
            self.data_path = '/content/drive/My Drive/' + 'train_images/'
        elif self.subset == "test":
            self.data_path = '/content/drive/My Drive/' + 'train_images/'
        self.on_epoch_end()

    def __len__(self):
        return int(np.floor(len(self.df) / self.batch_size))

    def on_epoch_end(self):
        self.indexes = np.arange(len(self.df))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __getitem__(self, index):
        train_datagen = ImageDataGenerator()
        param = {'flip_horizontal':True, 'samplewise_std_normalization' : True}

        X = np.empty((self.batch_size,128,800,3),dtype=np.float32)
        y = np.empty((self.batch_size,128,800,4),dtype=np.int8)
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]
        for i,f in enumerate(self.df['ImageId'].iloc[indexes]):
```

```
        for i,f in enumerate(self.df['ImageId'].iloc[indexes]):
            self.info[index*self.batch_size+i]=f
            img = Image.open(self.data_path + f).resize((800,128))
            X[i,] = train_datagen.apply_transform(x = img, transform_parameters = param)
            if self.subset == 'train':
                for j in range(4):
                    mask = rle2mask(self.df['e'+str(j+1)].iloc[indexes[i]])
                    y[i,:,:,j] = train_datagen.apply_transform(x = mask, transform_parameters = param)
        if self.preprocess!=None: X = self.preprocess(X)
        if self.subset == 'train': return X, y
        else: return X
```

In [0]:

```
class DataGenerator2(keras.utils.Sequence):
    def __init__(self, df, batch_size = 16, subset="train", shuffle=False,
                 preprocess=None, info={}):
        super().__init__()
        self.df = df
        self.shuffle = shuffle
        self.subset = subset
        self.batch_size = batch_size
        self.preprocess = preprocess
        self.info = info

        if self.subset == "train":
            self.data_path = '/content/drive/My Drive/' + 'train_images/'
        elif self.subset == "test":
            self.data_path = '/content/drive/My Drive/' + 'train_images/'
        self.on_epoch_end()

    def __len__(self):
        return int(np.floor(len(self.df) / self.batch_size))

    def on_epoch_end(self):
        self.indexes = np.arange(len(self.df))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    def __getitem__(self, index):

        X = np.empty((self.batch_size,128,800,3),dtype=np.float32)
        y = np.empty((self.batch_size,128,800,4),dtype=np.int8)
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]
        for i,f in enumerate(self.df['ImageId'].iloc[indexes]):
            self.info[index*self.batch_size+i]=f
            img = Image.open(self.data_path + f).resize((800,128))
            X[i,] = img
            if self.subset == 'train':
                for j in range(4):
                    mask = rle2mask(self.df['e'+str(j+1)].iloc[indexes[i]])
                    y[i,:,:,j] = mask
        if self.preprocess!=None: X = self.preprocess(X)
        if self.subset == 'train': return X, y
        else: return X
```

In [0]:

```
# https://www.kaggle.com/xhlulu/severstal-simple-keras-u-net-boilerplate
from keras import backend as K
from keras.losses import binary_crossentropy
# Competetion Metric
def dice_coef(y_true, y_pred, smooth=1):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def  bce_dice_loss(y_true, y_predict):
  return binary_crossentropy(y_true, y_predict) + (1-dice_coef(y_true, y_predict))
```

Training Model1

Training Model

In [0]:

```python
# Model taken from https://www.kaggle.com/ateplyuk/keras-starter-u-net

inputs = Input((128, 800, 3))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c5)

u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u6)
c6 = Dropout(0.2) (c6)
c6 = Conv2D(128, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c6)

u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)
u7 = concatenate([u7, c3])
c7 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u7)
c7 = Dropout(0.2) (c7)
c7 = Conv2D(64, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c7)

u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c7)
u8 = concatenate([u8, c2])
c8 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u8)
c8 = Dropout(0.1) (c8)
c8 = Conv2D(32, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c8)

u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c8)
u9 = concatenate([u9, c1], axis=3)
c9 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (u9)
c9 = Dropout(0.1) (c9)
c9 = Conv2D(16, (3, 3), activation='elu', kernel_initializer='he_normal', padding='same') (c9)

outputs = Conv2D(4, (1, 1), activation='sigmoid') (c9)

model_n = Model(inputs=[inputs], outputs=[outputs])
model_n.compile(optimizer='adam', loss='binary_crossentropy' , metrics=['accuracy',dice_coef])
```

In [0]:

```python
# Fit model
train_batches = DataGenerator(train_df,shuffle=True)
valid_batches = DataGenerator(cv_df)
history = model_n.fit_generator(train_batches, validation_data = valid_batches, epochs = 20, verbos
e=1)
```

```
Epoch 1/20
628/628 [==============================] - 4875s 8s/step - loss: 0.0443 - acc: 0.9901 - dice_coef:
0.0293 - val_loss: 0.0379 - val_acc: 0.9928 - val_dice_coef: 0.0284
Epoch 2/20
628/628 [==============================] - 205s 327ms/step - loss: 0.0343 - acc: 0.9921 -
dice_coef: 0.0718 - val_loss: 0.0299 - val_acc: 0.9928 - val_dice_coef: 0.1291
```

```
Epoch 3/20
628/628 [==============================] - 209s 333ms/step - loss: 0.0307 - acc: 0.9923 -
dice_coef: 0.1325 - val_loss: 0.0262 - val_acc: 0.9932 - val_dice_coef: 0.1929
Epoch 4/20
628/628 [==============================] - 211s 336ms/step - loss: 0.0283 - acc: 0.9925 -
dice_coef: 0.1823 - val_loss: 0.0253 - val_acc: 0.9933 - val_dice_coef: 0.1989
Epoch 5/20
628/628 [==============================] - 211s 336ms/step - loss: 0.0258 - acc: 0.9929 -
dice_coef: 0.2305 - val_loss: 0.0241 - val_acc: 0.9936 - val_dice_coef: 0.2224
Epoch 6/20
628/628 [==============================] - 211s 337ms/step - loss: 0.0232 - acc: 0.9934 -
dice_coef: 0.2840 - val_loss: 0.0198 - val_acc: 0.9943 - val_dice_coef: 0.3033
Epoch 7/20
628/628 [==============================] - 211s 335ms/step - loss: 0.0218 - acc: 0.9936 -
dice_coef: 0.3098 - val_loss: 0.0185 - val_acc: 0.9945 - val_dice_coef: 0.3601
Epoch 8/20
628/628 [==============================] - 211s 336ms/step - loss: 0.0202 - acc: 0.9938 -
dice_coef: 0.3474 - val_loss: 0.0184 - val_acc: 0.9945 - val_dice_coef: 0.3368
Epoch 9/20
628/628 [==============================] - 211s 335ms/step - loss: 0.0193 - acc: 0.9939 -
dice_coef: 0.3650 - val_loss: 0.0178 - val_acc: 0.9946 - val_dice_coef: 0.4062
Epoch 10/20
628/628 [==============================] - 211s 335ms/step - loss: 0.0184 - acc: 0.9941 -
dice_coef: 0.3921 - val_loss: 0.0167 - val_acc: 0.9946 - val_dice_coef: 0.3904
Epoch 11/20
628/628 [==============================] - 210s 335ms/step - loss: 0.0173 - acc: 0.9943 -
dice_coef: 0.4112 - val_loss: 0.0150 - val_acc: 0.9950 - val_dice_coef: 0.4714
Epoch 12/20
628/628 [==============================] - 211s 336ms/step - loss: 0.0168 - acc: 0.9944 -
dice_coef: 0.4276 - val_loss: 0.0153 - val_acc: 0.9951 - val_dice_coef: 0.4744
Epoch 13/20
628/628 [==============================] - 210s 335ms/step - loss: 0.0157 - acc: 0.9947 -
dice_coef: 0.4565 - val_loss: 0.0146 - val_acc: 0.9951 - val_dice_coef: 0.4776
Epoch 14/20
628/628 [==============================] - 210s 335ms/step - loss: 0.0154 - acc: 0.9947 -
dice_coef: 0.4592 - val_loss: 0.0143 - val_acc: 0.9951 - val_dice_coef: 0.4786
Epoch 15/20
628/628 [==============================] - 210s 334ms/step - loss: 0.0150 - acc: 0.9948 -
dice_coef: 0.4734 - val_loss: 0.0145 - val_acc: 0.9952 - val_dice_coef: 0.4649
Epoch 16/20
628/628 [==============================] - 210s 334ms/step - loss: 0.0142 - acc: 0.9951 -
dice_coef: 0.4988 - val_loss: 0.0126 - val_acc: 0.9957 - val_dice_coef: 0.4854
Epoch 17/20
628/628 [==============================] - 209s 333ms/step - loss: 0.0137 - acc: 0.9952 -
dice_coef: 0.5083 - val_loss: 0.0136 - val_acc: 0.9953 - val_dice_coef: 0.5138
Epoch 18/20
628/628 [==============================] - 209s 333ms/step - loss: 0.0140 - acc: 0.9951 -
dice_coef: 0.5012 - val_loss: 0.0125 - val_acc: 0.9957 - val_dice_coef: 0.5350
Epoch 19/20
628/628 [==============================] - 210s 334ms/step - loss: 0.0130 - acc: 0.9954 -
dice_coef: 0.5305 - val_loss: 0.0124 - val_acc: 0.9957 - val_dice_coef: 0.5530
Epoch 20/20
628/628 [==============================] - 209s 333ms/step - loss: 0.0129 - acc: 0.9954 -
dice_coef: 0.5307 - val_loss: 0.0122 - val_acc: 0.9958 - val_dice_coef: 0.5441
```

In [0]:

```
from keras.models import load_model

model_n.save('my_model1.h5')
```

In [0]:

```
from google.colab import files
files.download( "/content/my_model1.h5" )
```

In [0]:

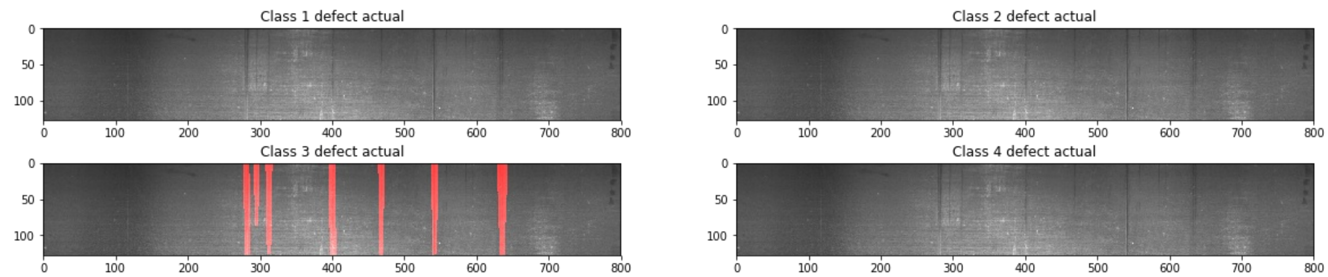
```
test_batches = DataGenerator2(test_df, subset='test',batch_size=1)
preds = model_n.predict_generator(test_batches,verbose=1)
```

```
1257/1257 [==============================] - 576s 458ms/step
```

Visyalizing the predicted value

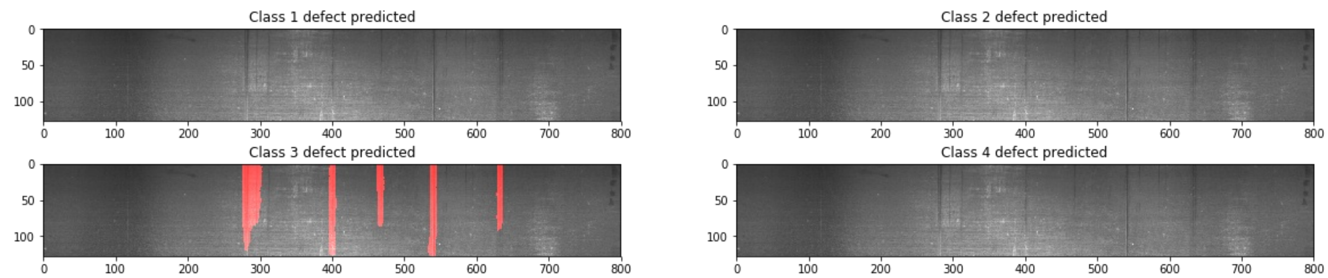
In [0]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[13]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[13])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
y_predicted = preds[13]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
# Predicting on test data
from tqdm import tqdm
data_path = '/content/drive/My Drive/' + 'test_images/'
files = list(os.listdir(data_path))
img_classId = []
rle_lst = []
for f in files:
  X = np.empty((1,128,800,3),dtype=np.float32)
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
```

```
  X[0,] = img
  mask = model_n.predict(X)
  #print(mask[0,:,:,1].shape)
  rle_m = np.empty((128,800),dtype=np.uint8)
  for i in range(4):
    rle_m = mask[0,:,:,i].round().astype(int)
    rle = mask2rle(rle_m)
    rle_lst.append(rle)
    img_classId.append(f+'_'+str(i+1))
```

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission1.csv', index=False)
```

In [0]:

```
from google.colab import files
files.download( "/content/submission1.csv" )
```

Training Model 2

In [0]:

```
inputs = Input((128,800,3))
s = Lambda(lambda x: x / 255) (inputs)

c1 = Conv2D(8, (3, 3), activation='elu', padding='same') (s)
c1 = Conv2D(8, (3, 3), activation='elu', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(16, (3, 3), activation='elu', padding='same') (p1)
c2 = Conv2D(16, (3, 3), activation='elu', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(32, (3, 3), activation='elu', padding='same') (p2)
c3 = Conv2D(32, (3, 3), activation='elu', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(64, (3, 3), activation='elu', padding='same') (p3)
c4 = Conv2D(64, (3, 3), activation='elu', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(64, (3, 3), activation='elu', padding='same') (p4)
c5 = Conv2D(64, (3, 3), activation='elu', padding='same') (c5)
p5 = MaxPooling2D(pool_size=(2, 2)) (c5)

c55 = Conv2D(128, (3, 3), activation='elu', padding='same') (p5)
c55 = Conv2D(128, (3, 3), activation='elu', padding='same') (c55)

u6 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c55)
u6 = concatenate([u6, c5])
c6 = Conv2D(64, (3, 3), activation='elu', padding='same') (u6)
c6 = Conv2D(64, (3, 3), activation='elu', padding='same') (c6)

u71 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c6)
u71 = concatenate([u71, c4])
c71 = Conv2D(32, (3, 3), activation='elu', padding='same') (u71)
c61 = Conv2D(32, (3, 3), activation='elu', padding='same') (c71)

u7 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c61)
u7 = concatenate([u7, c3])
c7 = Conv2D(32, (3, 3), activation='elu', padding='same') (u7)
c7 = Conv2D(32, (3, 3), activation='elu', padding='same') (c7)

u8 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c7)
u8 = concatenate([u8, c2])
c8 = Conv2D(16, (3, 3), activation='elu', padding='same') (u8)
c8 = Conv2D(16, (3, 3), activation='elu', padding='same') (c8)
```

```
c8 = Conv2D(16, (3, 3), activation='elu', padding='same') (c8)

u9 = Conv2DTranspose(8, (2, 2), strides=(2, 2), padding='same') (c8)
u9 = concatenate([u9, c1], axis=3)
c9 = Conv2D(8, (3, 3), activation='elu', padding='same') (u9)
c9 = Conv2D(8, (3, 3), activation='elu', padding='same') (c9)

outputs = Conv2D(4, (1, 1), activation='sigmoid') (c9)

model = Model(inputs=[inputs], outputs=[outputs])
model.compile(optimizer='adam', loss=bce_dice_loss, metrics=[dice_coef])
```

In [0]:

```
# Fit model
train_batches = DataGenerator(train_df,shuffle=True)
valid_batches = DataGenerator(cv_df)
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs = 20, verbose=
1)
```

```
Epoch 1/20
628/628 [==============================] - 160s 254ms/step - loss: 0.0565 - dice_coef: 0.0191 - va
l_loss: 0.0332 - val_dice_coef: 0.0275
Epoch 2/20
628/628 [==============================] - 151s 240ms/step - loss: 0.0340 - dice_coef: 0.0571 - va
l_loss: 0.0294 - val_dice_coef: 0.0727
Epoch 3/20
628/628 [==============================] - 152s 243ms/step - loss: 0.0286 - dice_coef: 0.1213 - va
l_loss: 0.0253 - val_dice_coef: 0.1654
Epoch 4/20
628/628 [==============================] - 151s 240ms/step - loss: 0.0236 - dice_coef: 0.2132 - va
l_loss: 0.0222 - val_dice_coef: 0.2016
Epoch 5/20
628/628 [==============================] - 150s 239ms/step - loss: 0.0215 - dice_coef: 0.2548 - va
l_loss: 0.0215 - val_dice_coef: 0.2658
Epoch 6/20
628/628 [==============================] - 151s 241ms/step - loss: 0.0188 - dice_coef: 0.2992 - va
l_loss: 0.0164 - val_dice_coef: 0.3089
Epoch 7/20
628/628 [==============================] - 151s 240ms/step - loss: 0.0167 - dice_coef: 0.3377 - va
l_loss: 0.0163 - val_dice_coef: 0.3118
Epoch 8/20
628/628 [==============================] - 152s 243ms/step - loss: 0.0152 - dice_coef: 0.3645 - va
l_loss: 0.0127 - val_dice_coef: 0.3684
Epoch 9/20
628/628 [==============================] - 154s 245ms/step - loss: 0.0146 - dice_coef: 0.3808 - va
l_loss: 0.0116 - val_dice_coef: 0.3897
Epoch 10/20
628/628 [==============================] - 152s 241ms/step - loss: 0.0130 - dice_coef: 0.4070 - va
l_loss: 0.0110 - val_dice_coef: 0.4658
Epoch 11/20
628/628 [==============================] - 151s 240ms/step - loss: 0.0113 - dice_coef: 0.4448 - va
l_loss: 0.0108 - val_dice_coef: 0.4577
Epoch 12/20
628/628 [==============================] - 152s 242ms/step - loss: 0.0112 - dice_coef: 0.4532 - va
l_loss: 0.0119 - val_dice_coef: 0.4026
Epoch 13/20
628/628 [==============================] - 153s 244ms/step - loss: 0.0110 - dice_coef: 0.4574 - va
l_loss: 0.0091 - val_dice_coef: 0.4794
Epoch 14/20
628/628 [==============================] - 154s 245ms/step - loss: 0.0101 - dice_coef: 0.4796 - va
l_loss: 0.0094 - val_dice_coef: 0.4848
Epoch 15/20
628/628 [==============================] - 158s 252ms/step - loss: 0.0093 - dice_coef: 0.4939 - va
l_loss: 0.0080 - val_dice_coef: 0.5088
Epoch 16/20
628/628 [==============================] - 160s 254ms/step - loss: 0.0091 - dice_coef: 0.5061 - va
l_loss: 0.0102 - val_dice_coef: 0.4761
Epoch 17/20
628/628 [==============================] - 155s 247ms/step - loss: 0.0086 - dice_coef: 0.5167 - va
l_loss: 0.0090 - val_dice_coef: 0.4829
Epoch 18/20
628/628 [==============================] - 153s 243ms/step - loss: 0.0082 - dice_coef: 0.5283 - va
l_loss: 0.0078 - val_dice_coef: 0.5508
Epoch 19/20
628/628 [==============================] - 154s 245ms/step - loss: 0.0081 - dice_coef: 0.5365 - va
```

```
l_loss: 0.0084 - val_dice_coef: 0.4965
Epoch 20/20
628/628 [==============================] - 162s 258ms/step - loss: 0.0074 - dice_coef: 0.5483 - va
l_loss: 0.0098 - val_dice_coef: 0.5028
```

In [0]:

```
from keras.models import load_model

model.save('my_model2.h5')
```

In [0]:

```
from google.colab import files
files.download( "/content/my_model2.h5" )
```

In [0]:

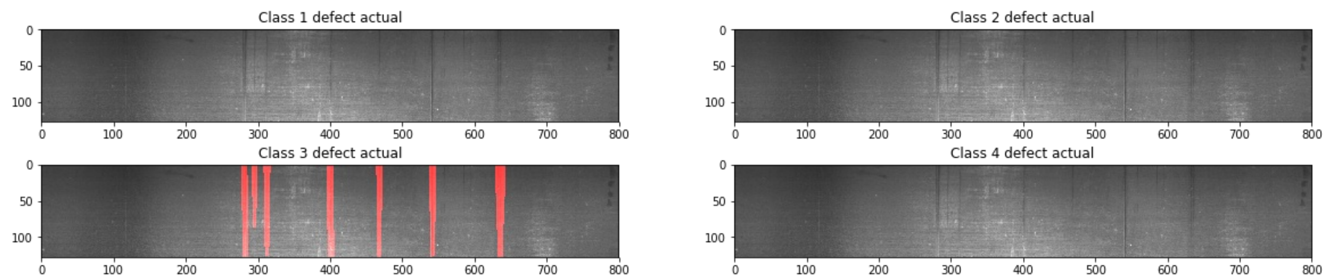
```
test_batches = DataGenerator2(test_df, subset='test',batch_size=1)
preds = model.predict_generator(test_batches,verbose=1)
```

```
1257/1257 [==============================] - 21s 17ms/step
```

Tesing the model and visualizing

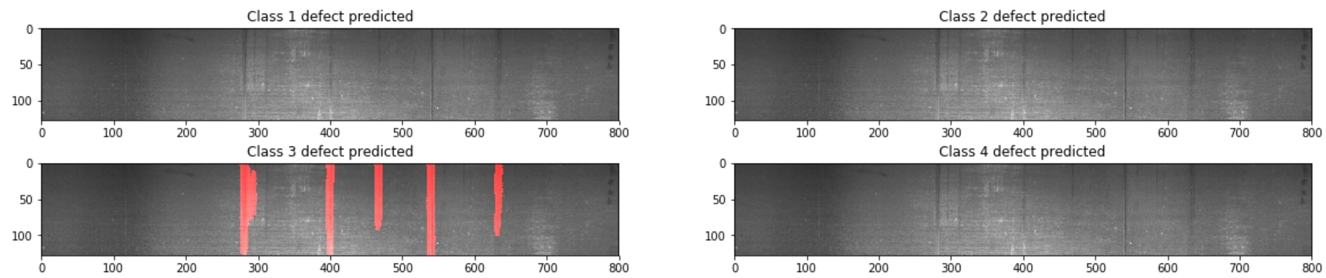
In [0]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[13]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[13])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
y_predicted = preds[13]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
# Predicting on test data
from tqdm import tqdm
data_path = '/content/drive/My Drive/' + 'test_images/'
files = list(os.listdir(data_path))
img_classId = []
rle_lst = []
for f in files:
  X = np.empty((1,128,800,3),dtype=np.float32)
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  X[0,] = img
  mask = model_n.predict(X)
  #print(mask[0,:,:,1].shape)
  rle_m = np.empty((128,800),dtype=np.uint8)
  for i in range(4):
    rle_m = mask[0,:,:,i].round().astype(int)
    rle = mask2rle(rle_m)
    rle_lst.append(rle)
    img_classId.append(f+'_'+str(i+1))
```

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission2.csv',index=False)
```

In [0]:

```
from google.colab import files
files.download( "/content/submission2.csv" )
```

Using image segmentation model

In [15]:

```
! pip install segmentation-models
```

```
Collecting segmentation-models
  Downloading
https://files.pythonhosted.org/packages/10/bf/253c8834014a834cacf2384c72872167fb30ccae7a56c6ce46285
45c/segmentation_models-0.2.1-py2.py3-none-any.whl (44kB)
     |████████████████████████████████| 51kB 2.5MB/s
Requirement already satisfied: keras-applications>=1.0.7 in /usr/local/lib/python3.6/dist-packages
(from segmentation-models) (1.0.8)
Collecting image-classifiers==0.2.0 (from segmentation-models)
  Downloading
https://files.pythonhosted.org/packages/de/32/a1e74e03f74506d1e4b46bb2732ca5a7b18ac52a36b5e3547e635
74c/image_classifiers-0.2.0-py2.py3-none-any.whl (76kB)
     |████████████████████████████████| 81kB 7.5MB/s
Requirement already satisfied: keras>=2.2.0 in /usr/local/lib/python3.6/dist-packages (from
segmentation-models) (2.2.5)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist-packages (from
segmentation-models) (0.15.0)
```

```
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-applications>=1.0.7->segmentation-models) (2.8.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from keras-applications>=1.0.7->segmentation-models) (1.16.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-models) (3.13)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-models) (1.12.0)
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-models) (1.1.0)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-models) (1.3.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentation-models) (2.3)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentation-models) (4.3.0)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentation-models) (3.0.3)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentation-models) (1.0.3)
Requirement already satisfied: imageio>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentation-models) (2.4.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx>=2.0->scikit-image->segmentation-models) (4.4.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-packages (from pillow>=4.3.0->scikit-image->segmentation-models) (0.46)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->segmentation-models) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->segmentation-models) (2.4.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->segmentation-models) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->segmentation-models) (2.5.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->segmentation-models) (41.2.0)
Installing collected packages: image-classifiers, segmentation-models
Successfully installed image-classifiers-0.2.0 segmentation-models-0.2.1
```

Using vgg16 as backbone

In [0]:

```
from segmentation_models import Unet
from segmentation_models.backbones import get_preprocessing

# LOAD UNET WITH PRETRAINING FROM IMAGENET
preprocess = get_preprocessing('vgg16') # for resnet, img = (img-110.0)/1.0
model2 = Unet('vgg16', input_shape=(128, 800, 3), classes=4, activation='sigmoid')
model2.compile(optimizer='adam', loss= bce_dice_loss, metrics=[dice_coef])
model2.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 4s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is deprecated. P
```

```
packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is deprecated. P
lease use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please us
e tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:2239: The name tf.image.resize_nearest_neighbor is de
precated. Please use tf.compat.v1.image.resize_nearest_neighbor instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is deprecated.
Please use tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "u-vgg16"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 128, 800, 3)  0
__________________________________________________________________________________________________
block1_conv1 (Conv2D)           (None, 128, 800, 64) 1792        input_1[0][0]
__________________________________________________________________________________________________
block1_conv2 (Conv2D)           (None, 128, 800, 64) 36928       block1_conv1[0][0]
__________________________________________________________________________________________________
block1_pool (MaxPooling2D)      (None, 64, 400, 64)  0           block1_conv2[0][0]
__________________________________________________________________________________________________
block2_conv1 (Conv2D)           (None, 64, 400, 128) 73856       block1_pool[0][0]
__________________________________________________________________________________________________
block2_conv2 (Conv2D)           (None, 64, 400, 128) 147584      block2_conv1[0][0]
__________________________________________________________________________________________________
block2_pool (MaxPooling2D)      (None, 32, 200, 128) 0           block2_conv2[0][0]
__________________________________________________________________________________________________
block3_conv1 (Conv2D)           (None, 32, 200, 256) 295168      block2_pool[0][0]
__________________________________________________________________________________________________
block3_conv2 (Conv2D)           (None, 32, 200, 256) 590080      block3_conv1[0][0]
__________________________________________________________________________________________________
block3_conv3 (Conv2D)           (None, 32, 200, 256) 590080      block3_conv2[0][0]
__________________________________________________________________________________________________
block3_pool (MaxPooling2D)      (None, 16, 100, 256) 0           block3_conv3[0][0]
__________________________________________________________________________________________________
block4_conv1 (Conv2D)           (None, 16, 100, 512) 1180160     block3_pool[0][0]
__________________________________________________________________________________________________
block4_conv2 (Conv2D)           (None, 16, 100, 512) 2359808     block4_conv1[0][0]
__________________________________________________________________________________________________
block4_conv3 (Conv2D)           (None, 16, 100, 512) 2359808     block4_conv2[0][0]
__________________________________________________________________________________________________
block4_pool (MaxPooling2D)      (None, 8, 50, 512)   0           block4_conv3[0][0]
__________________________________________________________________________________________________
block5_conv1 (Conv2D)           (None, 8, 50, 512)   2359808     block4_pool[0][0]
__________________________________________________________________________________________________
block5_conv2 (Conv2D)           (None, 8, 50, 512)   2359808     block5_conv1[0][0]
__________________________________________________________________________________________________
block5_conv3 (Conv2D)           (None, 8, 50, 512)   2359808     block5_conv2[0][0]
__________________________________________________________________________________________________
block5_pool (MaxPooling2D)      (None, 4, 25, 512)   0           block5_conv3[0][0]
__________________________________________________________________________________________________
decoder_stage0_upsample (UpSamp (None, 8, 50, 512)   0           block5_pool[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 8, 50, 1024)  0           decoder_stage0_upsample[0][0]
                                                                 block5_conv3[0][0]
__________________________________________________________________________________________________
decoder_stage0_conv1 (Conv2D)   (None, 8, 50, 256)   2359296     concatenate_1[0][0]
__________________________________________________________________________________________________
decoder_stage0_bn1 (BatchNormal (None, 8, 50, 256)   1024        decoder_stage0_conv1[0][0]
__________________________________________________________________________________________________
decoder_stage0_relu1 (Activatio (None, 8, 50, 256)   0           decoder_stage0_bn1[0][0]
__________________________________________________________________________________________________
decoder_stage0_conv2 (Conv2D)   (None, 8, 50, 256)   589824      decoder_stage0_relu1[0][0]
```

```
________________________________________________________________________________________________
decoder_stage0_bn2 (BatchNormal (None, 8, 50, 256)   1024        decoder_stage0_conv2[0][0]
________________________________________________________________________________________________
decoder_stage0_relu2 (Activatio (None, 8, 50, 256)   0           decoder_stage0_bn2[0][0]
________________________________________________________________________________________________
decoder_stage1_upsample (UpSamp (None, 16, 100, 256) 0           decoder_stage0_relu2[0][0]
________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 16, 100, 768) 0           decoder_stage1_upsample[0][0]
                                                                 block4_conv3[0][0]
________________________________________________________________________________________________
decoder_stage1_conv1 (Conv2D)   (None, 16, 100, 128) 884736      concatenate_2[0][0]
________________________________________________________________________________________________
decoder_stage1_bn1 (BatchNormal (None, 16, 100, 128) 512         decoder_stage1_conv1[0][0]
________________________________________________________________________________________________
decoder_stage1_relu1 (Activatio (None, 16, 100, 128) 0           decoder_stage1_bn1[0][0]
________________________________________________________________________________________________
decoder_stage1_conv2 (Conv2D)   (None, 16, 100, 128) 147456      decoder_stage1_relu1[0][0]
________________________________________________________________________________________________
decoder_stage1_bn2 (BatchNormal (None, 16, 100, 128) 512         decoder_stage1_conv2[0][0]
________________________________________________________________________________________________
decoder_stage1_relu2 (Activatio (None, 16, 100, 128) 0           decoder_stage1_bn2[0][0]
________________________________________________________________________________________________
decoder_stage2_upsample (UpSamp (None, 32, 200, 128) 0           decoder_stage1_relu2[0][0]
________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 32, 200, 384) 0           decoder_stage2_upsample[0][0]
                                                                 block3_conv3[0][0]
________________________________________________________________________________________________
decoder_stage2_conv1 (Conv2D)   (None, 32, 200, 64)  221184      concatenate_3[0][0]
________________________________________________________________________________________________
decoder_stage2_bn1 (BatchNormal (None, 32, 200, 64)  256         decoder_stage2_conv1[0][0]
________________________________________________________________________________________________
decoder_stage2_relu1 (Activatio (None, 32, 200, 64)  0           decoder_stage2_bn1[0][0]
________________________________________________________________________________________________
decoder_stage2_conv2 (Conv2D)   (None, 32, 200, 64)  36864       decoder_stage2_relu1[0][0]
________________________________________________________________________________________________
decoder_stage2_bn2 (BatchNormal (None, 32, 200, 64)  256         decoder_stage2_conv2[0][0]
________________________________________________________________________________________________
decoder_stage2_relu2 (Activatio (None, 32, 200, 64)  0           decoder_stage2_bn2[0][0]
________________________________________________________________________________________________
decoder_stage3_upsample (UpSamp (None, 64, 400, 64)  0           decoder_stage2_relu2[0][0]
________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 64, 400, 192) 0           decoder_stage3_upsample[0][0]
                                                                 block2_conv2[0][0]
________________________________________________________________________________________________
decoder_stage3_conv1 (Conv2D)   (None, 64, 400, 32)  55296       concatenate_4[0][0]
________________________________________________________________________________________________
decoder_stage3_bn1 (BatchNormal (None, 64, 400, 32)  128         decoder_stage3_conv1[0][0]
________________________________________________________________________________________________
decoder_stage3_relu1 (Activatio (None, 64, 400, 32)  0           decoder_stage3_bn1[0][0]
________________________________________________________________________________________________
decoder_stage3_conv2 (Conv2D)   (None, 64, 400, 32)  9216        decoder_stage3_relu1[0][0]
________________________________________________________________________________________________
decoder_stage3_bn2 (BatchNormal (None, 64, 400, 32)  128         decoder_stage3_conv2[0][0]
________________________________________________________________________________________________
decoder_stage3_relu2 (Activatio (None, 64, 400, 32)  0           decoder_stage3_bn2[0][0]
________________________________________________________________________________________________
decoder_stage4_upsample (UpSamp (None, 128, 800, 32) 0           decoder_stage3_relu2[0][0]
________________________________________________________________________________________________
decoder_stage4_conv1 (Conv2D)   (None, 128, 800, 16) 4608        decoder_stage4_upsample[0][0]
________________________________________________________________________________________________
decoder_stage4_bn1 (BatchNormal (None, 128, 800, 16) 64          decoder_stage4_conv1[0][0]
________________________________________________________________________________________________
decoder_stage4_relu1 (Activatio (None, 128, 800, 16) 0           decoder_stage4_bn1[0][0]
________________________________________________________________________________________________
decoder_stage4_conv2 (Conv2D)   (None, 128, 800, 16) 2304        decoder_stage4_relu1[0][0]
________________________________________________________________________________________________
decoder_stage4_bn2 (BatchNormal (None, 128, 800, 16) 64          decoder_stage4_conv2[0][0]
________________________________________________________________________________________________
decoder_stage4_relu2 (Activatio (None, 128, 800, 16) 0           decoder_stage4_bn2[0][0]
________________________________________________________________________________________________
final_conv (Conv2D)             (None, 128, 800, 4)  580         decoder_stage4_relu2[0][0]
________________________________________________________________________________________________
sigmoid (Activation)            (None, 128, 800, 4)  0           final_conv[0][0]
================================================================================================
Total params: 19,030,020
Trainable params: 19,028,036
Non-trainable params: 1,984
```

In [0]:

```
# TRAIN AND VALIDATE MODEL
train_batches = DataGenerator(train_df,shuffle=True,preprocess=preprocess)
valid_batches = DataGenerator(cv_df,preprocess=preprocess)
history = model2.fit_generator(train_batches, validation_data = valid_batches, epochs = 30, verbose
=1)
```

```
Epoch 1/30
628/628 [==============================] - 493s 785ms/step - loss: 0.5927 - dice_coef: 0.4499 - va
l_loss: 0.5689 - val_dice_coef: 0.4725
Epoch 2/30
628/628 [==============================] - 502s 799ms/step - loss: 0.5439 - dice_coef: 0.4957 - va
l_loss: 0.5050 - val_dice_coef: 0.5283
Epoch 3/30
628/628 [==============================] - 501s 798ms/step - loss: 0.5220 - dice_coef: 0.5153 - va
l_loss: 0.4931 - val_dice_coef: 0.5371
Epoch 4/30
628/628 [==============================] - 501s 798ms/step - loss: 0.5010 - dice_coef: 0.5356 - va
l_loss: 0.4850 - val_dice_coef: 0.5510
Epoch 5/30
628/628 [==============================] - 499s 795ms/step - loss: 0.4899 - dice_coef: 0.5460 - va
l_loss: 0.5179 - val_dice_coef: 0.5153
Epoch 6/30
628/628 [==============================] - 499s 795ms/step - loss: 0.4736 - dice_coef: 0.5608 - va
l_loss: 0.4604 - val_dice_coef: 0.5715
Epoch 7/30
628/628 [==============================] - 499s 794ms/step - loss: 0.4480 - dice_coef: 0.5849 - va
l_loss: 0.4277 - val_dice_coef: 0.6009
Epoch 8/30
628/628 [==============================] - 498s 793ms/step - loss: 0.4292 - dice_coef: 0.6019 - va
l_loss: 0.4712 - val_dice_coef: 0.5583
Epoch 9/30
628/628 [==============================] - 498s 793ms/step - loss: 0.4163 - dice_coef: 0.6142 - va
l_loss: 0.4012 - val_dice_coef: 0.6257
Epoch 10/30
628/628 [==============================] - 498s 793ms/step - loss: 0.4019 - dice_coef: 0.6277 - va
l_loss: 0.4038 - val_dice_coef: 0.6245
Epoch 11/30
628/628 [==============================] - 497s 792ms/step - loss: 0.3815 - dice_coef: 0.6465 - va
l_loss: 0.4098 - val_dice_coef: 0.6173
Epoch 12/30
628/628 [==============================] - 498s 793ms/step - loss: 0.3780 - dice_coef: 0.6495 - va
l_loss: 0.3763 - val_dice_coef: 0.6492
Epoch 13/30
628/628 [==============================] - 498s 792ms/step - loss: 0.3581 - dice_coef: 0.6682 - va
l_loss: 0.3890 - val_dice_coef: 0.6375
Epoch 14/30
628/628 [==============================] - 497s 792ms/step - loss: 0.3597 - dice_coef: 0.6671 - va
l_loss: 0.4024 - val_dice_coef: 0.6234
Epoch 15/30
628/628 [==============================] - 497s 792ms/step - loss: 0.3387 - dice_coef: 0.6866 - va
l_loss: 0.4489 - val_dice_coef: 0.5823
Epoch 16/30
628/628 [==============================] - 497s 792ms/step - loss: 0.3293 - dice_coef: 0.6952 - va
l_loss: 0.3803 - val_dice_coef: 0.6466
Epoch 17/30
628/628 [==============================] - 498s 793ms/step - loss: 0.3206 - dice_coef: 0.7035 - va
l_loss: 0.4194 - val_dice_coef: 0.6109
Epoch 18/30
628/628 [==============================] - 498s 793ms/step - loss: 0.3069 - dice_coef: 0.7162 - va
l_loss: 0.4226 - val_dice_coef: 0.6077
Epoch 19/30
628/628 [==============================] - 498s 793ms/step - loss: 0.3024 - dice_coef: 0.7205 - va
l_loss: 0.3811 - val_dice_coef: 0.6453
Epoch 20/30
628/628 [==============================] - 498s 793ms/step - loss: 0.3077 - dice_coef: 0.7155 - va
l_loss: 0.3935 - val_dice_coef: 0.6349
Epoch 21/30
628/628 [==============================] - 498s 793ms/step - loss: 0.2860 - dice_coef: 0.7357 - va
l_loss: 0.3743 - val_dice_coef: 0.6513
Epoch 22/30
628/628 [==============================] - 498s 793ms/step - loss: 0.2742 - dice_coef: 0.7465 - va
l_loss: 0.3548 - val_dice_coef: 0.6697
```

```
Epoch 23/30
628/628 [==============================] - 498s 794ms/step - loss: 0.2731 - dice_coef: 0.7474 - va
l_loss: 0.3614 - val_dice_coef: 0.6649
Epoch 24/30
628/628 [==============================] - 498s 794ms/step - loss: 0.2613 - dice_coef: 0.7585 - va
l_loss: 0.3633 - val_dice_coef: 0.6637
Epoch 25/30
628/628 [==============================] - 499s 794ms/step - loss: 0.2536 - dice_coef: 0.7658 - va
l_loss: 0.3540 - val_dice_coef: 0.6714
Epoch 26/30
628/628 [==============================] - 499s 794ms/step - loss: 0.2488 - dice_coef: 0.7700 - va
l_loss: 0.3554 - val_dice_coef: 0.6698
Epoch 27/30
628/628 [==============================] - 498s 793ms/step - loss: 0.2509 - dice_coef: 0.7681 - va
l_loss: 0.3915 - val_dice_coef: 0.6382
Epoch 28/30
628/628 [==============================] - 498s 794ms/step - loss: 0.2410 - dice_coef: 0.7772 - va
l_loss: 0.3636 - val_dice_coef: 0.6626
Epoch 29/30
628/628 [==============================] - 498s 793ms/step - loss: 0.2358 - dice_coef: 0.7819 - va
l_loss: 0.3964 - val_dice_coef: 0.6335
Epoch 30/30
628/628 [==============================] - 498s 793ms/step - loss: 0.2309 - dice_coef: 0.7866 - va
l_loss: 0.3834 - val_dice_coef: 0.6449
```

In [0]:

```python
from keras.models import load_model
model2.save("/content/drive/My Drive/Project/my_model2.h5")
```

In [0]:

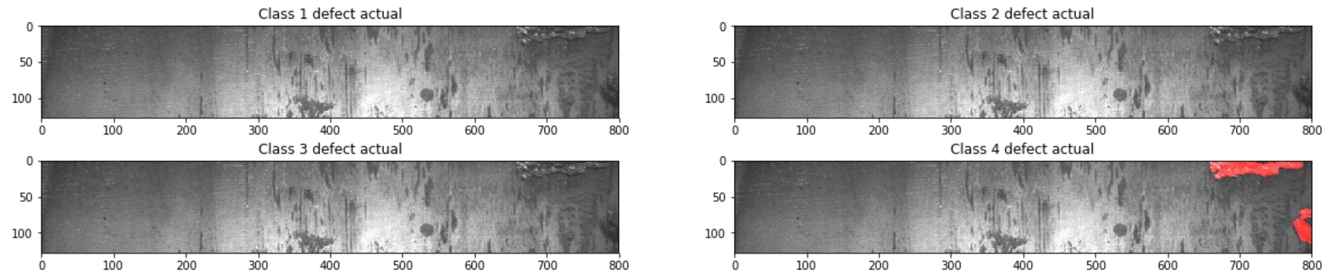
```python
test_batches = DataGenerator2(test_df,preprocess=preprocess, batch_size=1, subset='test')
preds = model2.predict_generator(test_batches,verbose=1)
```

```
1257/1257 [==============================] - 763s 607ms/step
```

Cases where model worked well

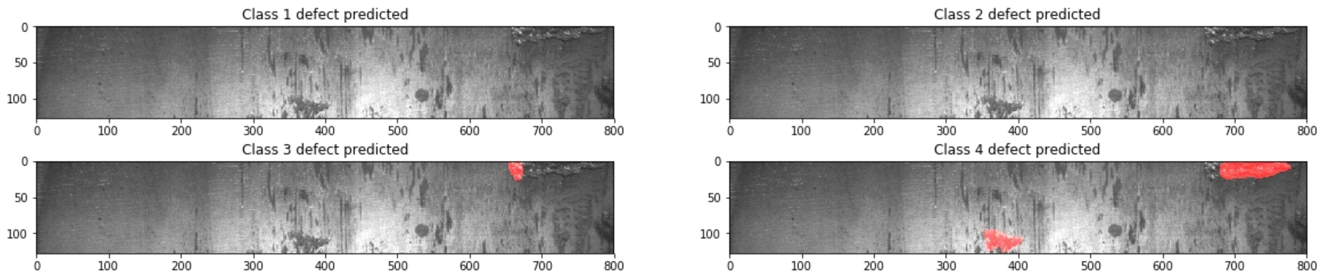
In [0]:

```python
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[2]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[2])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



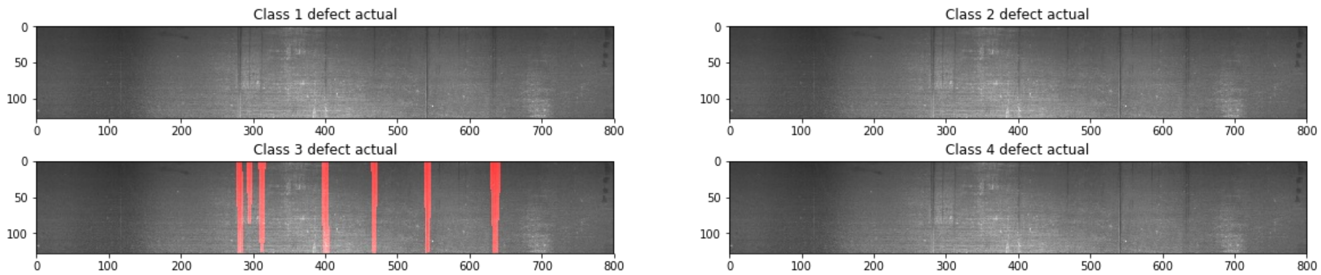
In [0]:

```python
y_predicted = preds[2]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



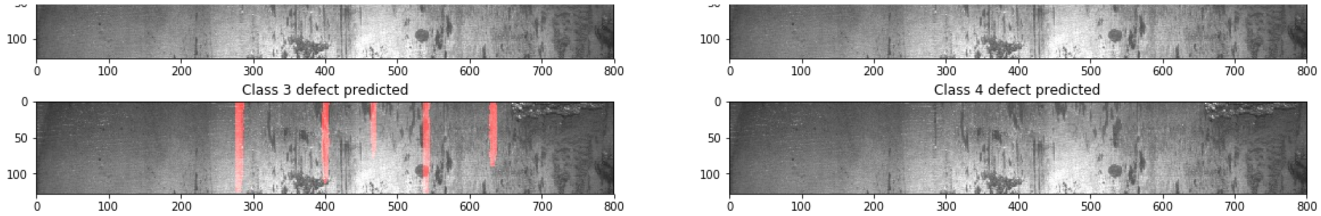
In [0]:

```python
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[13]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[13])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



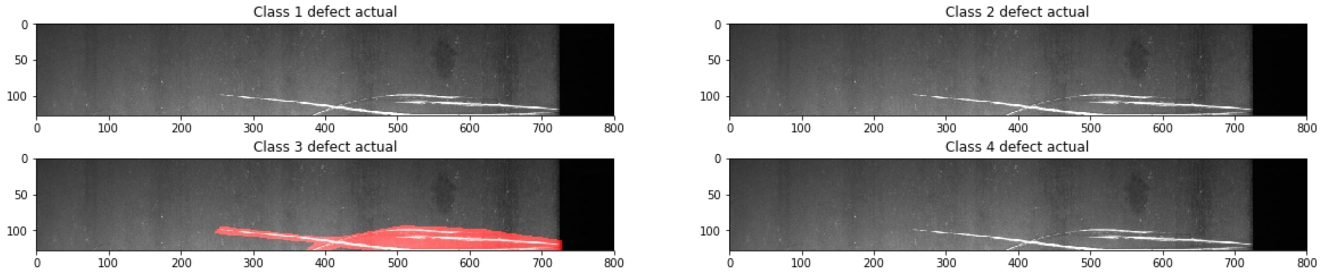
In [0]:

```python
y_predicted = preds[13]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



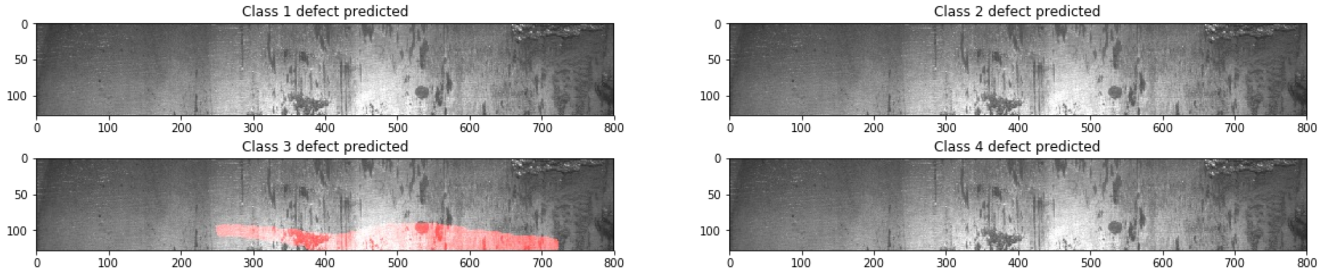
In [0]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[19]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[19])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
y_predicted = preds[19]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```

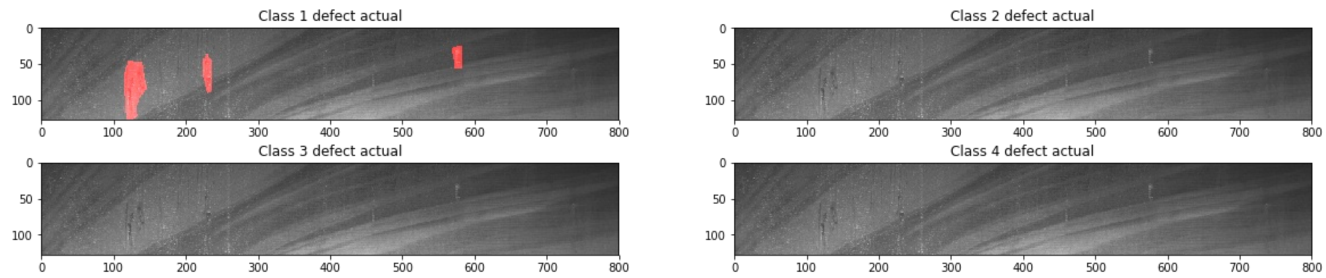


Cases where it failed

In [0]:

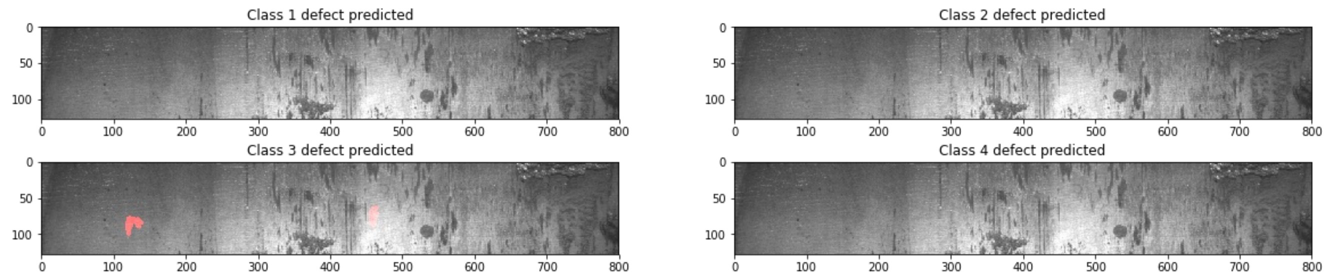
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
```

```
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[5]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[5])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
y_predicted = preds[5]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



Using restnet34 as backbone

In [0]:

```
from segmentation_models import Unet
from segmentation_models.backbones import get_preprocessing

# LOAD UNET WITH PRETRAINING FROM IMAGENET
preprocess = get_preprocessing('resnet34') # for resnet, img = (img-110.0)/1.0
model3 = Unet('resnet34', input_shape=(128, 800, 3), classes=4, activation='sigmoid')
model3.compile(optimizer='adam', loss= bce_dice_loss, metrics=[dice_coef])
model3.summary()
```

```
Downloading data from
https://github.com/qubvel/classification_models/releases/download/0.0.1/resnet34_imagenet_1000_no_t
5
85524480/85521592 [==============================] - 14s 0us/step
Model: "u-resnet34"
```

Layer (type)	Output Shape	Param #	Connected to
data (InputLayer)	(None, 128, 800, 3)	0	
bn_data (BatchNormalization)	(None, 128, 800, 3)	9	data[0][0]
zero_padding2d_1 (ZeroPadding2D	(None, 134, 806, 3)	0	bn_data[0][0]
conv0 (Conv2D)	(None, 64, 400, 64)	9408	zero_padding2d_1[0][0]
bn0 (BatchNormalization)	(None, 64, 400, 64)	256	conv0[0][0]
relu0 (Activation)	(None, 64, 400, 64)	0	bn0[0][0]
zero_padding2d_2 (ZeroPadding2D	(None, 66, 402, 64)	0	relu0[0][0]
pooling0 (MaxPooling2D)	(None, 32, 200, 64)	0	zero_padding2d_2[0][0]
stage1_unit1_bn1 (BatchNormaliz	(None, 32, 200, 64)	256	pooling0[0][0]
stage1_unit1_relu1 (Activation)	(None, 32, 200, 64)	0	stage1_unit1_bn1[0][0]
zero_padding2d_3 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit1_relu1[0][0]
stage1_unit1_conv1 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_3[0][0]
stage1_unit1_bn2 (BatchNormaliz	(None, 32, 200, 64)	256	stage1_unit1_conv1[0][0]
stage1_unit1_relu2 (Activation)	(None, 32, 200, 64)	0	stage1_unit1_bn2[0][0]
zero_padding2d_4 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit1_relu2[0][0]
stage1_unit1_conv2 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_4[0][0]
stage1_unit1_sc (Conv2D)	(None, 32, 200, 64)	4096	stage1_unit1_relu1[0][0]
add_1 (Add)	(None, 32, 200, 64)	0	stage1_unit1_conv2[0][0] stage1_unit1_sc[0][0]
stage1_unit2_bn1 (BatchNormaliz	(None, 32, 200, 64)	256	add_1[0][0]
stage1_unit2_relu1 (Activation)	(None, 32, 200, 64)	0	stage1_unit2_bn1[0][0]
zero_padding2d_5 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit2_relu1[0][0]
stage1_unit2_conv1 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_5[0][0]
stage1_unit2_bn2 (BatchNormaliz	(None, 32, 200, 64)	256	stage1_unit2_conv1[0][0]
stage1_unit2_relu2 (Activation)	(None, 32, 200, 64)	0	stage1_unit2_bn2[0][0]
zero_padding2d_6 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit2_relu2[0][0]
stage1_unit2_conv2 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_6[0][0]
add_2 (Add)	(None, 32, 200, 64)	0	stage1_unit2_conv2[0][0] add_1[0][0]
stage1_unit3_bn1 (BatchNormaliz	(None, 32, 200, 64)	256	add_2[0][0]
stage1_unit3_relu1 (Activation)	(None, 32, 200, 64)	0	stage1_unit3_bn1[0][0]
zero_padding2d_7 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit3_relu1[0][0]
stage1_unit3_conv1 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_7[0][0]
stage1_unit3_bn2 (BatchNormaliz	(None, 32, 200, 64)	256	stage1_unit3_conv1[0][0]
stage1_unit3_relu2 (Activation)	(None, 32, 200, 64)	0	stage1_unit3_bn2[0][0]
zero_padding2d_8 (ZeroPadding2D	(None, 34, 202, 64)	0	stage1_unit3_relu2[0][0]
stage1_unit3_conv2 (Conv2D)	(None, 32, 200, 64)	36864	zero_padding2d_8[0][0]
add_3 (Add)	(None, 32, 200, 64)	0	stage1_unit3_conv2[0][0] add_2[0][0]

stage2_unit1_bn1 (BatchNormaliz	(None, 32, 200, 64)	256	add_3[0][0]
stage2_unit1_relu1 (Activation)	(None, 32, 200, 64)	0	stage2_unit1_bn1[0][0]
zero_padding2d_9 (ZeroPadding2D	(None, 34, 202, 64)	0	stage2_unit1_relu1[0][0]
stage2_unit1_conv1 (Conv2D)	(None, 16, 100, 128)	73728	zero_padding2d_9[0][0]
stage2_unit1_bn2 (BatchNormaliz	(None, 16, 100, 128)	512	stage2_unit1_conv1[0][0]
stage2_unit1_relu2 (Activation)	(None, 16, 100, 128)	0	stage2_unit1_bn2[0][0]
zero_padding2d_10 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit1_relu2[0][0]
stage2_unit1_conv2 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_10[0][0]
stage2_unit1_sc (Conv2D)	(None, 16, 100, 128)	8192	stage2_unit1_relu1[0][0]
add_4 (Add)	(None, 16, 100, 128)	0	stage2_unit1_conv2[0][0] stage2_unit1_sc[0][0]
stage2_unit2_bn1 (BatchNormaliz	(None, 16, 100, 128)	512	add_4[0][0]
stage2_unit2_relu1 (Activation)	(None, 16, 100, 128)	0	stage2_unit2_bn1[0][0]
zero_padding2d_11 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit2_relu1[0][0]
stage2_unit2_conv1 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_11[0][0]
stage2_unit2_bn2 (BatchNormaliz	(None, 16, 100, 128)	512	stage2_unit2_conv1[0][0]
stage2_unit2_relu2 (Activation)	(None, 16, 100, 128)	0	stage2_unit2_bn2[0][0]
zero_padding2d_12 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit2_relu2[0][0]
stage2_unit2_conv2 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_12[0][0]
add_5 (Add)	(None, 16, 100, 128)	0	stage2_unit2_conv2[0][0] add_4[0][0]
stage2_unit3_bn1 (BatchNormaliz	(None, 16, 100, 128)	512	add_5[0][0]
stage2_unit3_relu1 (Activation)	(None, 16, 100, 128)	0	stage2_unit3_bn1[0][0]
zero_padding2d_13 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit3_relu1[0][0]
stage2_unit3_conv1 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_13[0][0]
stage2_unit3_bn2 (BatchNormaliz	(None, 16, 100, 128)	512	stage2_unit3_conv1[0][0]
stage2_unit3_relu2 (Activation)	(None, 16, 100, 128)	0	stage2_unit3_bn2[0][0]
zero_padding2d_14 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit3_relu2[0][0]
stage2_unit3_conv2 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_14[0][0]
add_6 (Add)	(None, 16, 100, 128)	0	stage2_unit3_conv2[0][0] add_5[0][0]
stage2_unit4_bn1 (BatchNormaliz	(None, 16, 100, 128)	512	add_6[0][0]
stage2_unit4_relu1 (Activation)	(None, 16, 100, 128)	0	stage2_unit4_bn1[0][0]
zero_padding2d_15 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit4_relu1[0][0]
stage2_unit4_conv1 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_15[0][0]
stage2_unit4_bn2 (BatchNormaliz	(None, 16, 100, 128)	512	stage2_unit4_conv1[0][0]
stage2_unit4_relu2 (Activation)	(None, 16, 100, 128)	0	stage2_unit4_bn2[0][0]
zero_padding2d_16 (ZeroPadding2	(None, 18, 102, 128)	0	stage2_unit4_relu2[0][0]
stage2_unit4_conv2 (Conv2D)	(None, 16, 100, 128)	147456	zero_padding2d_16[0][0]
add_7 (Add)	(None, 16, 100, 128)	0	stage2_unit4_conv2[0][0] add_6[0][0]

			add_6[0][0]
stage3_unit1_bn1 (BatchNormaliz	(None, 16, 100, 128)	512	add_7[0][0]
stage3_unit1_relu1 (Activation)	(None, 16, 100, 128)	0	stage3_unit1_bn1[0][0]
zero_padding2d_17 (ZeroPadding2	(None, 18, 102, 128)	0	stage3_unit1_relu1[0][0]
stage3_unit1_conv1 (Conv2D)	(None, 8, 50, 256)	294912	zero_padding2d_17[0][0]
stage3_unit1_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit1_conv1[0][0]
stage3_unit1_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit1_bn2[0][0]
zero_padding2d_18 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit1_relu2[0][0]
stage3_unit1_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_18[0][0]
stage3_unit1_sc (Conv2D)	(None, 8, 50, 256)	32768	stage3_unit1_relu1[0][0]
add_8 (Add)	(None, 8, 50, 256)	0	stage3_unit1_conv2[0][0] stage3_unit1_sc[0][0]
stage3_unit2_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_8[0][0]
stage3_unit2_relu1 (Activation)	(None, 8, 50, 256)	0	stage3_unit2_bn1[0][0]
zero_padding2d_19 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit2_relu1[0][0]
stage3_unit2_conv1 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_19[0][0]
stage3_unit2_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit2_conv1[0][0]
stage3_unit2_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit2_bn2[0][0]
zero_padding2d_20 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit2_relu2[0][0]
stage3_unit2_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_20[0][0]
add_9 (Add)	(None, 8, 50, 256)	0	stage3_unit2_conv2[0][0] add_8[0][0]
stage3_unit3_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_9[0][0]
stage3_unit3_relu1 (Activation)	(None, 8, 50, 256)	0	stage3_unit3_bn1[0][0]
zero_padding2d_21 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit3_relu1[0][0]
stage3_unit3_conv1 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_21[0][0]
stage3_unit3_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit3_conv1[0][0]
stage3_unit3_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit3_bn2[0][0]
zero_padding2d_22 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit3_relu2[0][0]
stage3_unit3_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_22[0][0]
add_10 (Add)	(None, 8, 50, 256)	0	stage3_unit3_conv2[0][0] add_9[0][0]
stage3_unit4_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_10[0][0]
stage3_unit4_relu1 (Activation)	(None, 8, 50, 256)	0	stage3_unit4_bn1[0][0]
zero_padding2d_23 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit4_relu1[0][0]
stage3_unit4_conv1 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_23[0][0]
stage3_unit4_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit4_conv1[0][0]
stage3_unit4_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit4_bn2[0][0]
zero_padding2d_24 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit4_relu2[0][0]
stage3_unit4_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_24[0][0]
add_11 (Add)	(None, 8, 50, 256)	0	stage3_unit4_conv2[0][0]

add_11 (Add)	(None, 8, 50, 256)	0	stage3_unit4_conv2[0][0] add_10[0][0]
stage3_unit5_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_11[0][0]
stage3_unit5_relu1 (Activation)	(None, 8, 50, 256)	0	stage3_unit5_bn1[0][0]
zero_padding2d_25 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit5_relu1[0][0]
stage3_unit5_conv1 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_25[0][0]
stage3_unit5_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit5_conv1[0][0]
stage3_unit5_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit5_bn2[0][0]
zero_padding2d_26 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit5_relu2[0][0]
stage3_unit5_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_26[0][0]
add_12 (Add)	(None, 8, 50, 256)	0	stage3_unit5_conv2[0][0] add_11[0][0]
stage3_unit6_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_12[0][0]
stage3_unit6_relu1 (Activation)	(None, 8, 50, 256)	0	stage3_unit6_bn1[0][0]
zero_padding2d_27 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit6_relu1[0][0]
stage3_unit6_conv1 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_27[0][0]
stage3_unit6_bn2 (BatchNormaliz	(None, 8, 50, 256)	1024	stage3_unit6_conv1[0][0]
stage3_unit6_relu2 (Activation)	(None, 8, 50, 256)	0	stage3_unit6_bn2[0][0]
zero_padding2d_28 (ZeroPadding2	(None, 10, 52, 256)	0	stage3_unit6_relu2[0][0]
stage3_unit6_conv2 (Conv2D)	(None, 8, 50, 256)	589824	zero_padding2d_28[0][0]
add_13 (Add)	(None, 8, 50, 256)	0	stage3_unit6_conv2[0][0] add_12[0][0]
stage4_unit1_bn1 (BatchNormaliz	(None, 8, 50, 256)	1024	add_13[0][0]
stage4_unit1_relu1 (Activation)	(None, 8, 50, 256)	0	stage4_unit1_bn1[0][0]
zero_padding2d_29 (ZeroPadding2	(None, 10, 52, 256)	0	stage4_unit1_relu1[0][0]
stage4_unit1_conv1 (Conv2D)	(None, 4, 25, 512)	1179648	zero_padding2d_29[0][0]
stage4_unit1_bn2 (BatchNormaliz	(None, 4, 25, 512)	2048	stage4_unit1_conv1[0][0]
stage4_unit1_relu2 (Activation)	(None, 4, 25, 512)	0	stage4_unit1_bn2[0][0]
zero_padding2d_30 (ZeroPadding2	(None, 6, 27, 512)	0	stage4_unit1_relu2[0][0]
stage4_unit1_conv2 (Conv2D)	(None, 4, 25, 512)	2359296	zero_padding2d_30[0][0]
stage4_unit1_sc (Conv2D)	(None, 4, 25, 512)	131072	stage4_unit1_relu1[0][0]
add_14 (Add)	(None, 4, 25, 512)	0	stage4_unit1_conv2[0][0] stage4_unit1_sc[0][0]
stage4_unit2_bn1 (BatchNormaliz	(None, 4, 25, 512)	2048	add_14[0][0]
stage4_unit2_relu1 (Activation)	(None, 4, 25, 512)	0	stage4_unit2_bn1[0][0]
zero_padding2d_31 (ZeroPadding2	(None, 6, 27, 512)	0	stage4_unit2_relu1[0][0]
stage4_unit2_conv1 (Conv2D)	(None, 4, 25, 512)	2359296	zero_padding2d_31[0][0]
stage4_unit2_bn2 (BatchNormaliz	(None, 4, 25, 512)	2048	stage4_unit2_conv1[0][0]
stage4_unit2_relu2 (Activation)	(None, 4, 25, 512)	0	stage4_unit2_bn2[0][0]
zero_padding2d_32 (ZeroPadding2	(None, 6, 27, 512)	0	stage4_unit2_relu2[0][0]
stage4_unit2_conv2 (Conv2D)	(None, 4, 25, 512)	2359296	zero_padding2d_32[0][0]

add_15 (Add)	(None, 4, 25, 512)	0	stage4_unit2_conv2[0][0] add_14[0][0]
stage4_unit3_bn1 (BatchNormaliz	(None, 4, 25, 512)	2048	add_15[0][0]
stage4_unit3_relu1 (Activation)	(None, 4, 25, 512)	0	stage4_unit3_bn1[0][0]
zero_padding2d_33 (ZeroPadding2	(None, 6, 27, 512)	0	stage4_unit3_relu1[0][0]
stage4_unit3_conv1 (Conv2D)	(None, 4, 25, 512)	2359296	zero_padding2d_33[0][0]
stage4_unit3_bn2 (BatchNormaliz	(None, 4, 25, 512)	2048	stage4_unit3_conv1[0][0]
stage4_unit3_relu2 (Activation)	(None, 4, 25, 512)	0	stage4_unit3_bn2[0][0]
zero_padding2d_34 (ZeroPadding2	(None, 6, 27, 512)	0	stage4_unit3_relu2[0][0]
stage4_unit3_conv2 (Conv2D)	(None, 4, 25, 512)	2359296	zero_padding2d_34[0][0]
add_16 (Add)	(None, 4, 25, 512)	0	stage4_unit3_conv2[0][0] add_15[0][0]
bn1 (BatchNormalization)	(None, 4, 25, 512)	2048	add_16[0][0]
relu1 (Activation)	(None, 4, 25, 512)	0	bn1[0][0]
decoder_stage0_upsample (UpSamp	(None, 8, 50, 512)	0	relu1[0][0]
concatenate_5 (Concatenate)	(None, 8, 50, 768)	0	decoder_stage0_upsample[0][0] stage4_unit1_relu1[0][0]
decoder_stage0_conv1 (Conv2D)	(None, 8, 50, 256)	1769472	concatenate_5[0][0]
decoder_stage0_bn1 (BatchNormal	(None, 8, 50, 256)	1024	decoder_stage0_conv1[0][0]
decoder_stage0_relu1 (Activatio	(None, 8, 50, 256)	0	decoder_stage0_bn1[0][0]
decoder_stage0_conv2 (Conv2D)	(None, 8, 50, 256)	589824	decoder_stage0_relu1[0][0]
decoder_stage0_bn2 (BatchNormal	(None, 8, 50, 256)	1024	decoder_stage0_conv2[0][0]
decoder_stage0_relu2 (Activatio	(None, 8, 50, 256)	0	decoder_stage0_bn2[0][0]
decoder_stage1_upsample (UpSamp	(None, 16, 100, 256)	0	decoder_stage0_relu2[0][0]
concatenate_6 (Concatenate)	(None, 16, 100, 384)	0	decoder_stage1_upsample[0][0] stage3_unit1_relu1[0][0]
decoder_stage1_conv1 (Conv2D)	(None, 16, 100, 128)	442368	concatenate_6[0][0]
decoder_stage1_bn1 (BatchNormal	(None, 16, 100, 128)	512	decoder_stage1_conv1[0][0]
decoder_stage1_relu1 (Activatio	(None, 16, 100, 128)	0	decoder_stage1_bn1[0][0]
decoder_stage1_conv2 (Conv2D)	(None, 16, 100, 128)	147456	decoder_stage1_relu1[0][0]
decoder_stage1_bn2 (BatchNormal	(None, 16, 100, 128)	512	decoder_stage1_conv2[0][0]
decoder_stage1_relu2 (Activatio	(None, 16, 100, 128)	0	decoder_stage1_bn2[0][0]
decoder_stage2_upsample (UpSamp	(None, 32, 200, 128)	0	decoder_stage1_relu2[0][0]
concatenate_7 (Concatenate)	(None, 32, 200, 192)	0	decoder_stage2_upsample[0][0] stage2_unit1_relu1[0][0]
decoder_stage2_conv1 (Conv2D)	(None, 32, 200, 64)	110592	concatenate_7[0][0]
decoder_stage2_bn1 (BatchNormal	(None, 32, 200, 64)	256	decoder_stage2_conv1[0][0]
decoder_stage2_relu1 (Activatio	(None, 32, 200, 64)	0	decoder_stage2_bn1[0][0]
decoder_stage2_conv2 (Conv2D)	(None, 32, 200, 64)	36864	decoder_stage2_relu1[0][0]
decoder_stage2_bn2 (BatchNormal	(None, 32, 200, 64)	256	decoder_stage2_conv2[0][0]
decoder_stage2_relu2 (Activatio	(None, 32, 200, 64)	0	decoder_stage2_bn2[0][0]

```
________________________________________________________________________________________________
decoder_stage3_upsample (UpSamp (None, 64, 400, 64)  0           decoder_stage2_relu2[0][0]
________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 64, 400, 128) 0           decoder_stage3_upsample[0][0]
                                                                 relu0[0][0]
________________________________________________________________________________________________
decoder_stage3_conv1 (Conv2D)   (None, 64, 400, 32)  36864       concatenate_8[0][0]
________________________________________________________________________________________________
decoder_stage3_bn1 (BatchNormal (None, 64, 400, 32)  128         decoder_stage3_conv1[0][0]
________________________________________________________________________________________________
decoder_stage3_relu1 (Activatio (None, 64, 400, 32)  0           decoder_stage3_bn1[0][0]
________________________________________________________________________________________________
decoder_stage3_conv2 (Conv2D)   (None, 64, 400, 32)  9216        decoder_stage3_relu1[0][0]
________________________________________________________________________________________________
decoder_stage3_bn2 (BatchNormal (None, 64, 400, 32)  128         decoder_stage3_conv2[0][0]
________________________________________________________________________________________________
decoder_stage3_relu2 (Activatio (None, 64, 400, 32)  0           decoder_stage3_bn2[0][0]
________________________________________________________________________________________________
decoder_stage4_upsample (UpSamp (None, 128, 800, 32) 0           decoder_stage3_relu2[0][0]
________________________________________________________________________________________________
decoder_stage4_conv1 (Conv2D)   (None, 128, 800, 16) 4608        decoder_stage4_upsample[0][0]
________________________________________________________________________________________________
decoder_stage4_bn1 (BatchNormal (None, 128, 800, 16) 64          decoder_stage4_conv1[0][0]
________________________________________________________________________________________________
decoder_stage4_relu1 (Activatio (None, 128, 800, 16) 0           decoder_stage4_bn1[0][0]
________________________________________________________________________________________________
decoder_stage4_conv2 (Conv2D)   (None, 128, 800, 16) 2304        decoder_stage4_relu1[0][0]
________________________________________________________________________________________________
decoder_stage4_bn2 (BatchNormal (None, 128, 800, 16) 64          decoder_stage4_conv2[0][0]
________________________________________________________________________________________________
decoder_stage4_relu2 (Activatio (None, 128, 800, 16) 0           decoder_stage4_bn2[0][0]
________________________________________________________________________________________________
final_conv (Conv2D)             (None, 128, 800, 4)  580         decoder_stage4_relu2[0][0]
________________________________________________________________________________________________
sigmoid (Activation)            (None, 128, 800, 4)  0           final_conv[0][0]
================================================================================================
Total params: 24,456,589
Trainable params: 24,439,239
Non-trainable params: 17,350
________________________________________________________________________________________________
```

In [0]:

```
# TRAIN AND VALIDATE MODEL
train_batches = DataGenerator(train_df,shuffle=True,preprocess=preprocess)
valid_batches = DataGenerator(cv_df,preprocess=preprocess)
history = model3.fit_generator(train_batches, validation_data = valid_batches, epochs = 30, verbose
=1)
```

```
Epoch 1/30
628/628 [==============================] - 318s 506ms/step - loss: 0.6975 - dice_coef: 0.3617 - va
l_loss: 0.7048 - val_dice_coef: 0.3381
Epoch 2/30
628/628 [==============================] - 315s 501ms/step - loss: 0.5771 - dice_coef: 0.4649 - va
l_loss: 0.6261 - val_dice_coef: 0.4099
Epoch 3/30
628/628 [==============================] - 314s 500ms/step - loss: 0.5377 - dice_coef: 0.5017 - va
l_loss: 0.5179 - val_dice_coef: 0.5219
Epoch 4/30
628/628 [==============================] - 315s 502ms/step - loss: 0.5103 - dice_coef: 0.5274 - va
l_loss: 1.0113 - val_dice_coef: 0.1974
Epoch 5/30
628/628 [==============================] - 314s 500ms/step - loss: 0.5112 - dice_coef: 0.5259 - va
l_loss: 0.5543 - val_dice_coef: 0.4800
Epoch 6/30
628/628 [==============================] - 314s 500ms/step - loss: 0.4956 - dice_coef: 0.5401 - va
l_loss: 0.4936 - val_dice_coef: 0.5436
Epoch 7/30
628/628 [==============================] - 314s 500ms/step - loss: 0.4782 - dice_coef: 0.5559 - va
l_loss: 0.5310 - val_dice_coef: 0.5070
Epoch 8/30
628/628 [==============================] - 315s 501ms/step - loss: 0.4806 - dice_coef: 0.5535 - va
l_loss: 0.4992 - val_dice_coef: 0.5300
Epoch 9/30
628/628 [==============================] - 315s 501ms/step - loss: 0.4770 - dice_coef: 0.5575 - va
l_loss: 0.4608 - val_dice_coef: 0.5680
```

```
Epoch 10/30
628/628 [==============================] - 315s 502ms/step - loss: 0.4649 - dice_coef: 0.5687 - va
l_loss: 0.4661 - val_dice_coef: 0.5666
Epoch 11/30
628/628 [==============================] - 315s 502ms/step - loss: 0.4481 - dice_coef: 0.5839 - va
l_loss: 0.5415 - val_dice_coef: 0.4990
Epoch 12/30
628/628 [==============================] - 315s 501ms/step - loss: 0.4404 - dice_coef: 0.5917 - va
l_loss: 0.5569 - val_dice_coef: 0.4881
Epoch 13/30
628/628 [==============================] - 314s 500ms/step - loss: 0.4210 - dice_coef: 0.6092 - va
l_loss: 0.3864 - val_dice_coef: 0.6402
Epoch 14/30
628/628 [==============================] - 314s 499ms/step - loss: 0.4139 - dice_coef: 0.6163 - va
l_loss: 0.4347 - val_dice_coef: 0.5972
Epoch 15/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3916 - dice_coef: 0.6366 - va
l_loss: 0.4068 - val_dice_coef: 0.6223
Epoch 16/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3850 - dice_coef: 0.6428 - va
l_loss: 0.4611 - val_dice_coef: 0.5705
Epoch 17/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3744 - dice_coef: 0.6527 - va
l_loss: 0.4638 - val_dice_coef: 0.5705
Epoch 18/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3902 - dice_coef: 0.6382 - va
l_loss: 0.5306 - val_dice_coef: 0.5105
Epoch 19/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3654 - dice_coef: 0.6612 - va
l_loss: 0.4262 - val_dice_coef: 0.6021
Epoch 20/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3774 - dice_coef: 0.6498 - va
l_loss: 0.3748 - val_dice_coef: 0.6540
Epoch 21/30
628/628 [==============================] - 313s 499ms/step - loss: 0.3630 - dice_coef: 0.6633 - va
l_loss: 0.3547 - val_dice_coef: 0.6696
Epoch 22/30
628/628 [==============================] - 314s 501ms/step - loss: 0.3501 - dice_coef: 0.6751 - va
l_loss: 0.3861 - val_dice_coef: 0.6400
Epoch 23/30
628/628 [==============================] - 313s 499ms/step - loss: 0.3467 - dice_coef: 0.6782 - va
l_loss: 0.3563 - val_dice_coef: 0.6691
Epoch 24/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3357 - dice_coef: 0.6885 - va
l_loss: 0.3644 - val_dice_coef: 0.6607
Epoch 25/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3385 - dice_coef: 0.6858 - va
l_loss: 0.4690 - val_dice_coef: 0.5642
Epoch 26/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3273 - dice_coef: 0.6960 - va
l_loss: 0.5173 - val_dice_coef: 0.5287
Epoch 27/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3308 - dice_coef: 0.6929 - va
l_loss: 0.3609 - val_dice_coef: 0.6643
Epoch 28/30
628/628 [==============================] - 312s 497ms/step - loss: 0.3139 - dice_coef: 0.7083 - va
l_loss: 0.3791 - val_dice_coef: 0.6462
Epoch 29/30
628/628 [==============================] - 312s 498ms/step - loss: 0.3118 - dice_coef: 0.7109 - va
l_loss: 0.3687 - val_dice_coef: 0.6562
Epoch 30/30
628/628 [==============================] - 313s 498ms/step - loss: 0.3065 - dice_coef: 0.7153 - va
l_loss: 0.3883 - val_dice_coef: 0.6416
```

In [0]:

```
from keras.models import load_model
model3.save("/content/drive/My Drive/Project/my_model3.h5")
```

In [0]:

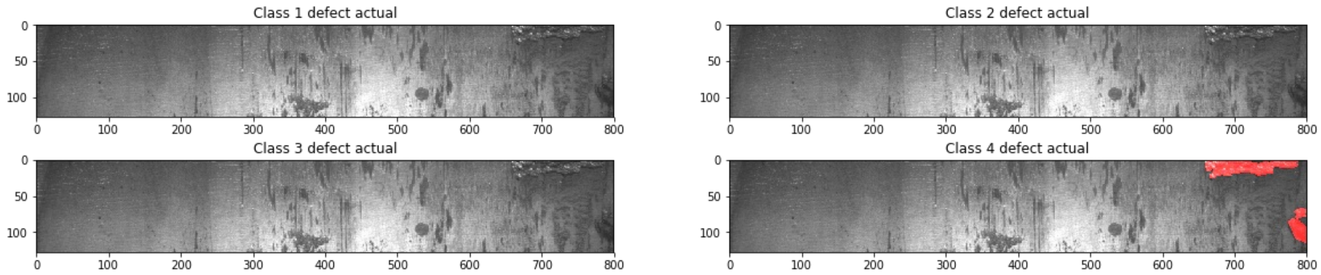
```
test_batches = DataGenerator2(test_df,preprocess=preprocess, subset='test')
preds = model3.predict_generator(test_batches,verbose=1)
```

```
78/78 [==============================] - 16s 209ms/step
```

Cases where the model worked

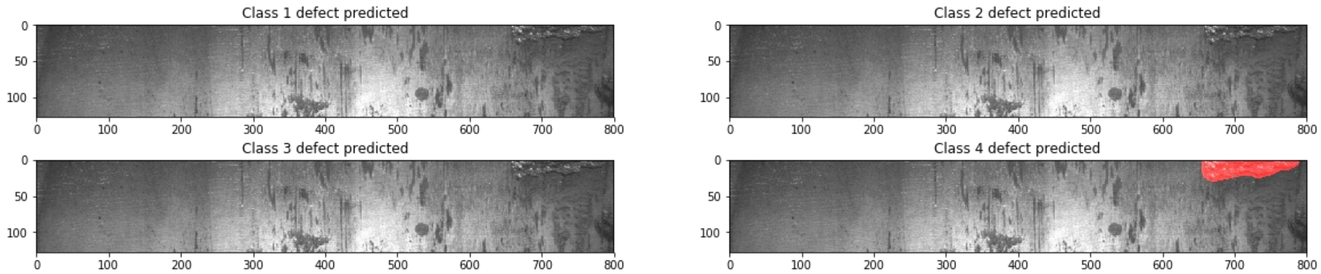
In [0]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[2]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[2])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

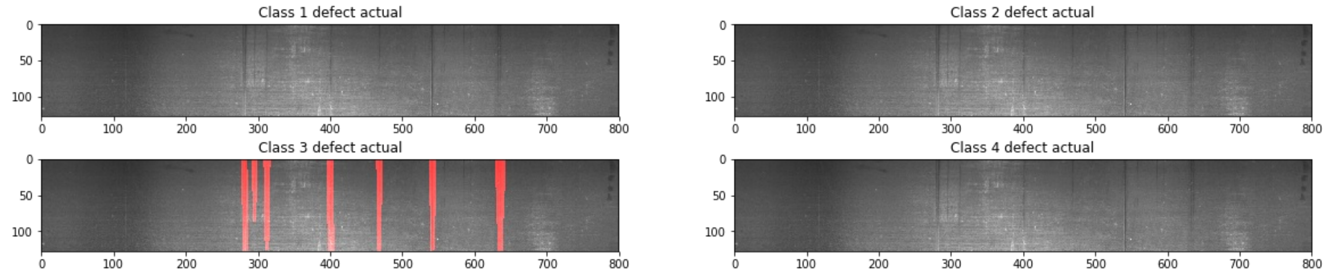
```
y_predicted = preds[2]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

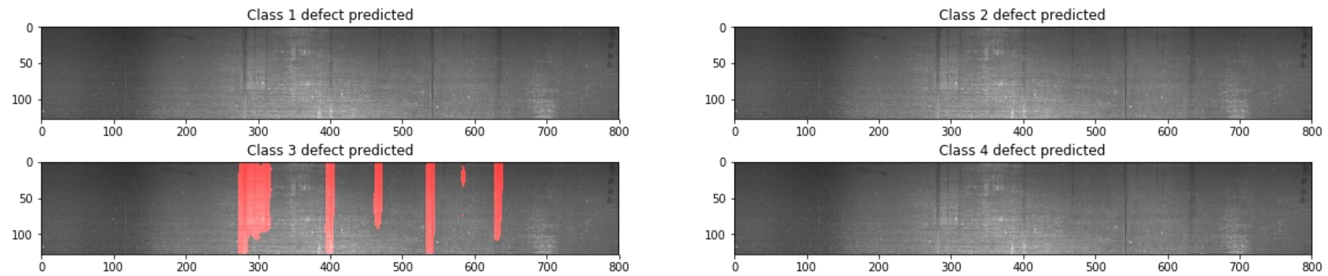
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[13]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[13])
```

```
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



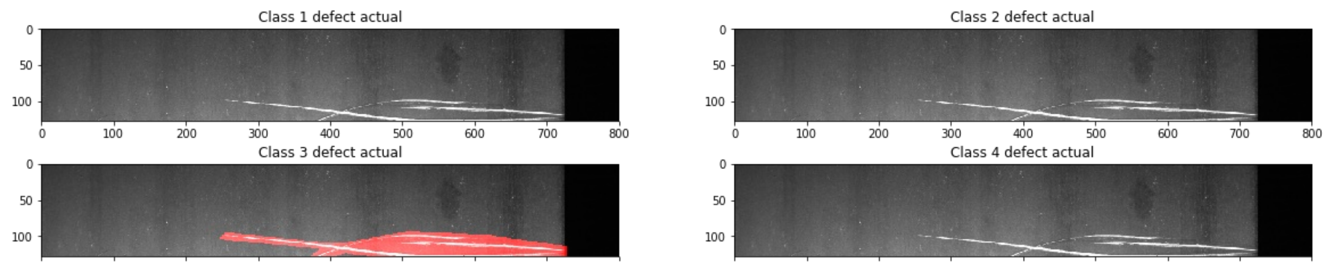
In [0]:

```
y_predicted = preds[13]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



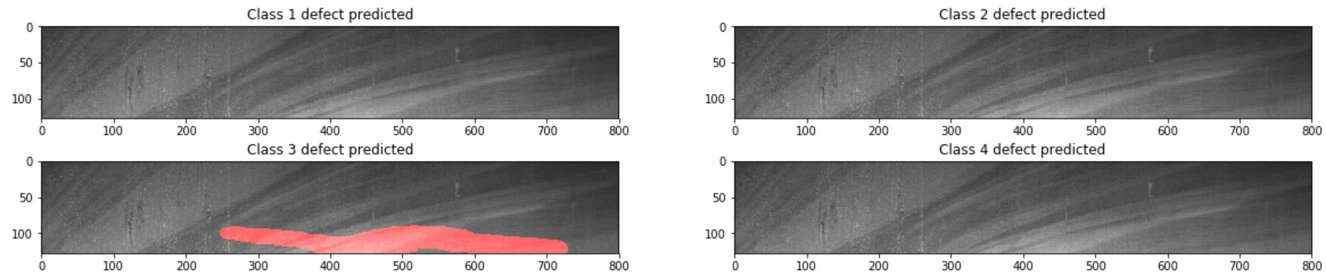
In [0]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[19]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[19])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

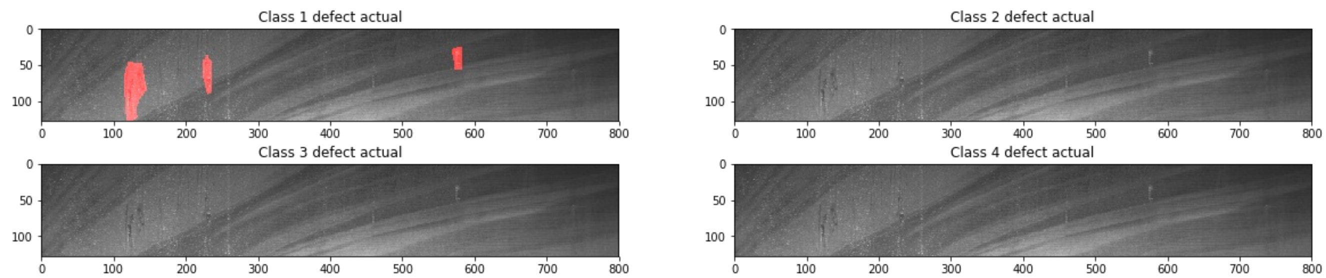
```
y_predicted = preds[19]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



Cases where the model failed

In [0]:

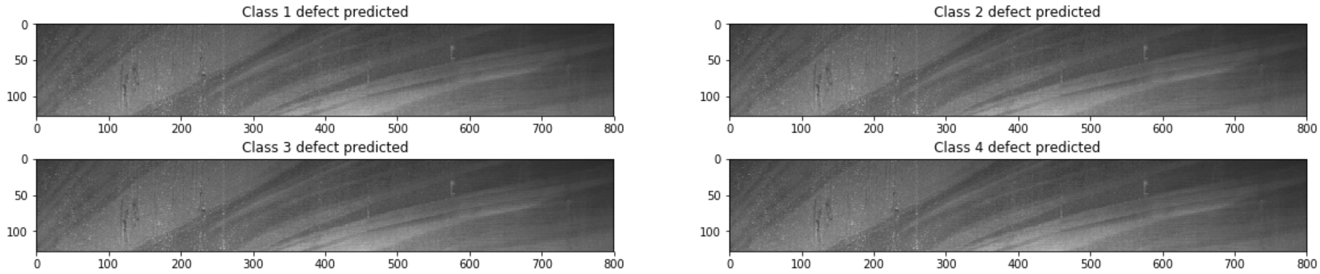
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,4))
data_path = '/content/drive/My Drive/' + 'train_images/'
f = test_df['ImageId'].iloc[5]
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = rle2mask( test_df['e'+str(i+1)].iloc[5])
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
  plt.title("Class {} defect actual".format(i+1))
  plt.imshow(img)

plt.show()
```



In [0]:

```
y_predicted = preds[5]
fig = plt.figure(figsize=(20,4))
for i in range(4):
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  mask = y_predicted[:,:,i].round().astype(int)
  img[mask==1,0] = 255
  fig.add_subplot(2, 2, i+1)
```

```
  plt.title("Class {} defect predicted".format(i+1))
  plt.imshow(img)

plt.show()
```



Predicting the output on test dataset and saving the output

In [0]:

```
# Predicting on training data
from tqdm import tqdm
data_path = '/content/drive/My Drive/' + 'test_images/'
files = list(os.listdir(data_path))
img_classId = []
rle_lst = []
for f in files:
  X = np.empty((1,128,800,3),dtype=np.float32)
  img = cv2.imread(data_path + f)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (800,128))
  X[0,] = img
  mask = model2.predict(X)
  #print(mask[0,:,:,1].shape)
  rle_m = np.empty((128,800),dtype=np.uint8)
  for i in range(4):
    rle_m = mask[0,:,:,i].round().astype(int)
    rle = mask2rle(rle_m)
    rle_lst.append(rle)
    img_classId.append(f+'_'+str(i+1))
```

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission.csv', index=False)
```

Conclusion

The best model was Unet with vgg16 as background which gave us a dice coefficient of 0.8127 on public leader board on kaggle.