Severstal: Steel Defect Detection

1.0 Business Problem

1.1 Description

Severstal is leading the charge in efficient steel mining and production. With over 50K employees, 5K clients, and 25K products, this company has produced 11.8 Million tonnes of steel, 4.7 Million tonnes of coal and 11 Million tonnes of iron core pellets in 2019 which justifies its dominance in the steel industry.

We know that steel is one of the most important and most widely used building materials of modern times due to its impressive properties like durability, thermal conductivity, resistance to natural and man-made wear and most importantly it is resistant to corrosion which makes the material ubiquitous around the world. In order to make steel production more efficient without compromising the quality, Severstal wants to leverage the advances of Artificial Intelligence like computer vision to identify defects in delicate flat steel sheets. These sheets are produced from a sequence of manufacturing processes that involves heating, rolling, drying and cutting

Off lately, Severstal is using images generated form high frequency cameras to power a defect detection algorithm and Severstal is expecting the AI engineers to improve the algorithm to detect defects with high precision.

Credits: Kaggle

Problem Statement

- Detect and localize the surface defects on a steel sheet provided the image of steel sheets.
- Classify the detected surface defects into one or multiple classes among class values [1, 2, 3, 4]

1.2 Sources/Useful Links

- Source : https://www.kaggle.com/c/severstal-steel-defect-detection/overview (https://www.kaggle.com/c/severstal-steel-defect-detection/overview)

Useful Links

- References :
- https://forums.fast.ai/t/understanding-the-dice-coefficient/5838 (https://forums.fast.ai/t/understanding-the-dice-coefficient/5838)
- https://www.kaggle.com/wh1tezzz/correct-dice-metrics-for-this-competition (https://www.kaggle.com/wh1tezzz/correct-dice-metrics-for-this-competition)
- https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/)
- https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda (https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda)
- https://www.kaggle.com/cdeotte/keras-unet-with-eda (https://www.kaggle.com/cdeotte/keras-unet-with-eda)
- https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly (https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly)

1.3 Real world/Business Objectives and Constraints

1. No strict latency concerns but defect identification and localization should not take a long time. In an ideal situation it is desirable to match with the frequency of cameras.
2. Interpretability can be obtained in the form of probability values for each of 4 class labels.

2.0 Machine Learning Probelm

2.1 Data

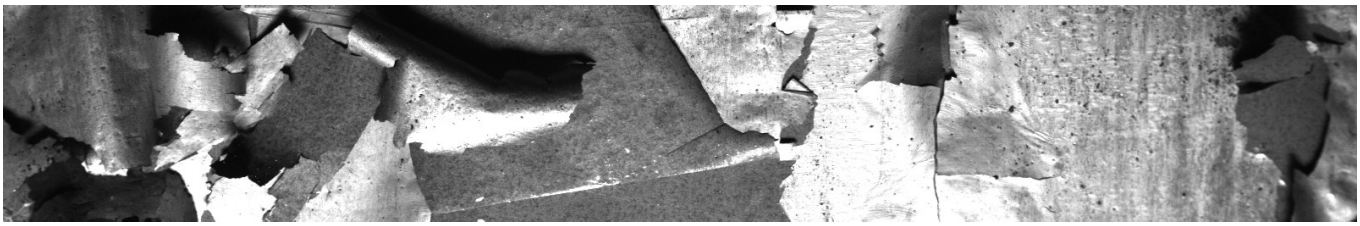
2.1.1 Data Overview

The data folder (size 2GB in total) provided by Severstal contains the following:

- train_images/ - folder of training images
- test_images/ - folder of test images (you are segmenting and classifying these images)
- train.csv - training annotations which provide segments for defects (ClassId = [1, 2, 3, 4])
- sample_submission.csv - a sample submission file in the correct format, with each ImageId repeated 4 times, one for each of the 4 defect classes

2.1.2 Example Data point

Sample train image



train_csv data point

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...

2.2 Mapping the real world problem to an ML problem

2.2.1 Type of Machine Leaning Problem

- This is an image segmentation and a classification problem to predict, localize the defects and then classify the detected defects

2.2.2 Performance Metric

- Dice coefficient.
- https://forums.fast.ai/t/understanding-the-dice-coefficient/5838 (https://forums.fast.ai/t/understanding-the-dice-coefficient/5838)
- https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient (https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient)

2.3 Train, CV and Test construction

- Training images will be split randomly into train & CV in the ratio 85:15.
- Testing images will be used for predictions.

3.0 Exploratory Data Analysis

Importing all the modules

In [0]:

```
import warnings
warnings.filterwarnings("ignore")
from datetime import datetime
import pandas as pd
import numpy as np
from numpy import asarray
import matplotlib.pyplot as plt
%matplotlib notebook
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import pickle
from tqdm import tqdm
import math
import random
import os
import cv2
from collections import Counter
from os import listdir
from matplotlib import image
from matplotlib import pyplot
from PIL import Image
from collections import defaultdict
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras import backend as K
from keras.losses import binary_crossentropy
from keras.layers import UpSampling2D, Conv2D, Activation, LeakyReLU, BatchNormalizatio
n,Input,Conv2DTranspose,Dropout
from keras.layers.pooling import MaxPooling2D, GlobalMaxPool2D
from keras import Model
from keras.preprocessing.image import ImageDataGenerator,array_to_img, img_to_array, lo
ad_img
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, Learning
RateScheduler,Callback
from keras.optimizers import Adam
from tqdm import tqdm_notebook
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate,add
```

The default version of TensorFlow in Colab will switch to TensorFlow 2.x on the 27th of March, 2020.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

```
Using TensorFlow backend.
```

3.1 Loading train csv file

In [0]:

```
#loading the train csv file containing pixels indicating defects
train_df= pd.read_csv("/content/train.csv")
train_df.head()
```

Out[0]:

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
3	000f6bf48.jpg	4	131973 1 132228 4 132483 6 132738 8 132993 11 ...
4	0014fce06.jpg	3	229501 11 229741 33 229981 55 230221 77 230468...

EncodedPixels columns indicate run-length encoding on the pixel values which are pairs of values that contain a start position and a run length. E.g. '1 3' implies starting at pixel 1 and running a total of 3 pixels (1,2,3).

The competition format requires a space delimited list of pairs. For example, '1 3 10 5' implies pixels 1,2,3,10,11,12,13,14 are to be included in the mask. The metric checks that the pairs are sorted, positive, and the decoded pixel values are not duplicated. The pixels are numbered from top to bottom, then left to right: 1 is pixel (1,1), 2 is pixel (2,1), etc.

In [0]:

```
print(train_df.shape)
```

```
(7095, 3)
```

- There are 7095 datapoints or steel sheet images containing defects

3.2 Checking for NaN

In [0]:

```
# train data
nan_rows = train_df[train_df.isnull().any(axis=1)]
nan_rows
```

Out[0]:

	ImageId	ClassId	EncodedPixels

- There are no NaN's

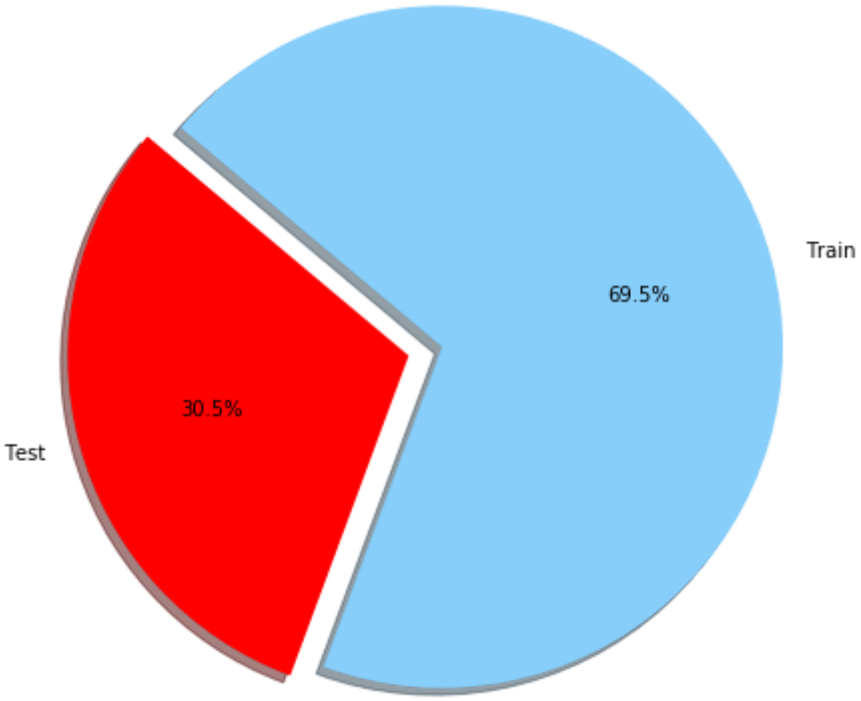
3.3 Analysing train & test image folders

3.3.1 Number of train & test images

In [0]:

```
train_count= 0
test_count= 0

for filename in listdir('train_images'):
    #counting number of images in train folder
    train_count+=1
for filename in listdir('test_images'):
    #counting number of images in test folder
    test_count+=1

print("Number of images in the train folder: ",train_count)
print("Number of images in the test folder: ",test_count)
print('-'*100)

#Pie-chart https://pythonspot.com/matplotlib-pie-chart/
# Data to plot
labels = 'Test', 'Train'
sizes = [test_count,train_count]
colors = ['red','lightskyblue']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.figure(figsize=(7,7))
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

```
Number of images in the train folder:  12568
Number of images in the test folder:  5506
--------------------------------------------------------------------------------
-------------------------
```



Looks like there is difference in image count in train folder & the imageID in train.csv

In [0]:

```
print("Number of images in the train folder which do not have defects are:",(train_coun
t-train_df.shape[0]))
```

```
Number of images in the train folder which do not have defects are: 5473
```

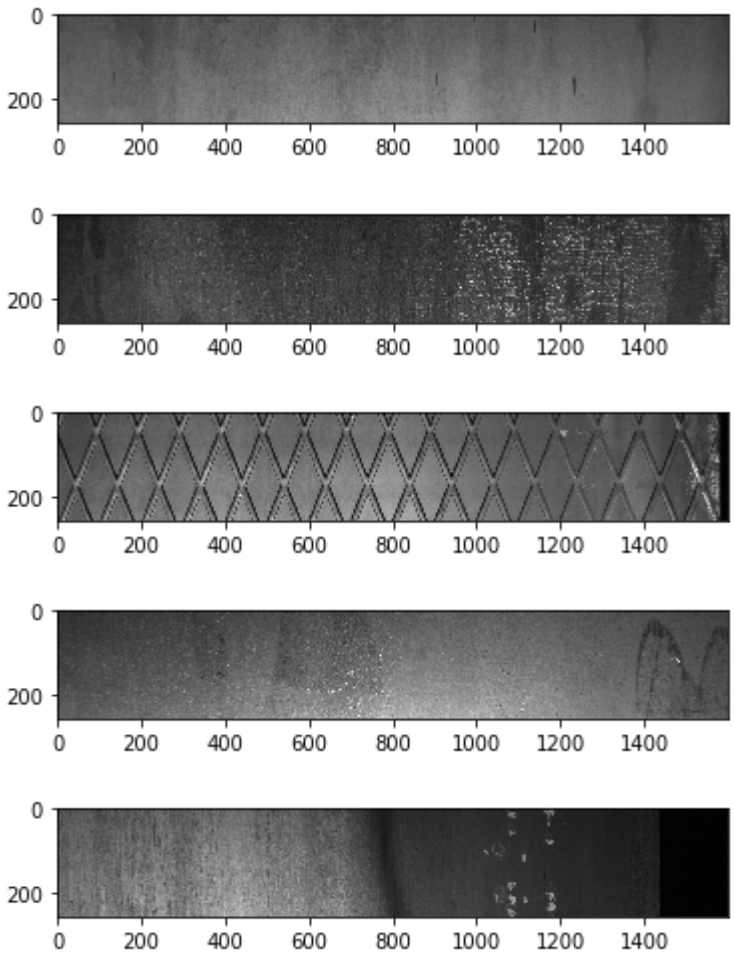
3.3.2 Printing few sample images containing no defects

In [0]:

```
# load all images in a directory
#from os import listdir
from matplotlib import image
# load all images in a directory
loaded_images = list()
defects= list(train_df.ImageId.values)
for filename in listdir('train_images'):
    if str(filename) not in defects:
# load image
        img_data = image.imread('train_images/' + filename)
# store loaded image
        loaded_images.append(img_data)
```

In [0]:

```
from matplotlib import pyplot
for i in range(5):
    disp= loaded_images[i]
    pyplot.imshow(disp)
    pyplot.show()
```



3.3.3 Basic properties of an image

In [0]:

```
from PIL import Image
# load the image
image = Image.open('0a5a82b86.jpg')
# summarize some details about the image
print(image.format)
print(image.mode)
print(image.size)
# show the image
image.show()
```

```
JPEG
RGB
(1600, 256)
```

3.3.4 Check if all images in train and test are of the same size

Train images

In [0]:

```
from PIL import Image
image_size=[]
for image_id in listdir('train_images'):
    img=Image.open("train_images/"+image_id)
    width,height=img.size
    image_size.append((width,height))
```

In [0]:

```
train_image_size_df=pd.DataFrame(image_size,columns=["width","height"])
train_image_size_df.head()
```

Out[0]:

	width	height
0	1600	256
1	1600	256
2	1600	256
3	1600	256
4	1600	256

In [0]:

```
print(train_image_size_df.width.unique())
print(train_image_size_df.height.unique())
```

```
[1600]
[256]
```

Test images

In [0]:

```
from PIL import Image
image_size=[]
for image_id in listdir('test_images'):
    img=Image.open("test_images/"+image_id)
    width,height=img.size
    image_size.append((width,height))
```

In [0]:

```
test_image_size_df=pd.DataFrame(image_size,columns=["width","height"])
test_image_size_df.head()
```

Out[0]:

	width	height
0	1600	256
1	1600	256
2	1600	256
3	1600	256
4	1600	256

In [0]:

```
print(test_image_size_df.width.unique())
print(test_image_size_df.height.unique())
```

```
[1600]
[256]
```

Therefore all images in train and test folders have the same size of (1600,256)

3.4 Analysis of response label: ClassId

3.4.1 Checking for class count

In [0]:

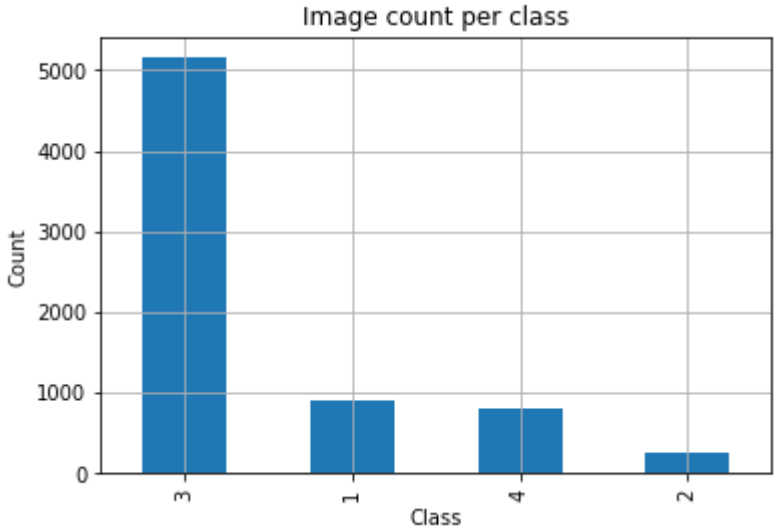
```
counts= train_df.ClassId.value_counts()
counts
```

Out[0]:

```
3    5150
1     897
4     801
2     247
Name: ClassId, dtype: int64
```

In [0]:

```
my_colors = 'rgbkymc'
counts.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Image count per class')
plt.grid()
plt.show()

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html
# -(train_class_distribution.values): the minus sign will give us in decreasing order
sorted_counts = np.argsort(-counts.values)
for i in sorted_counts:
    k=[3,1,4,2]
    print('Number of images in class', k[i], ':',counts.values[i], '(', np.round((count
s.values[i]/train_df.shape[0]*100), 3), '%)')
```




```
Number of images in class 3 : 5150 ( 72.586 %)
Number of images in class 1 : 897 ( 12.643 %)
Number of images in class 4 : 801 ( 11.29 %)
Number of images in class 2 : 247 ( 3.481 %)
```

Observations:

- Number of images with class 3 defect is found to be maximum compared to other defect classes
- Images with class 2 defects are least found
- Therefore the dataset is imbalanced with dominance of Class 3 defects**

3.4.1.1 Finding class weights

In [0]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html
from sklearn.utils import class_weight
class_wts = class_weight.compute_class_weight('balanced', np.unique(train_df.ClassId.values),train_df.ClassId)
print(class_wts)
for i in range(len(class_wts)):
    print("Weight for class {}: {}".format(i+1,class_wts[i]))
```

```
[1.97742475 7.18117409 0.34441748 2.21441948]
Weight for class 1: 1.9774247491638797
Weight for class 2: 7.181174089068826
Weight for class 3: 0.34441747572815534
Weight for class 4: 2.2144194756554305
```

Will be using these weights during training in the fit_generator function.

3.4.2 Checking number of labels tagged to each image

In [0]:

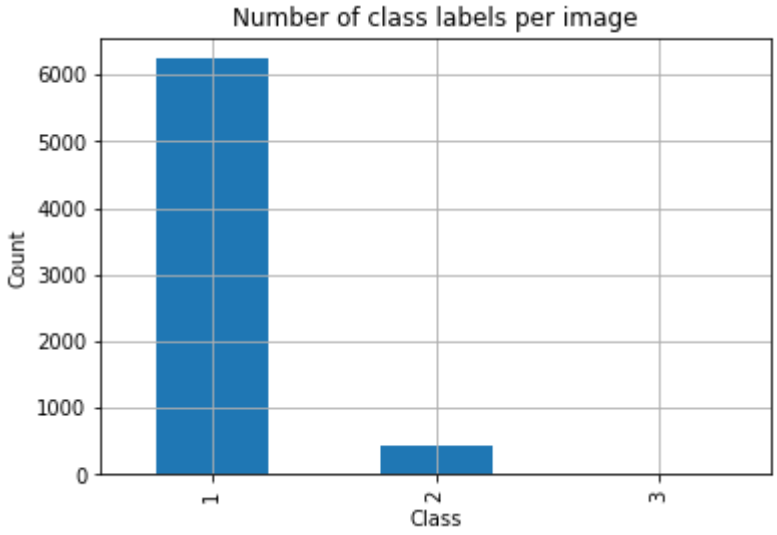
```
labels_per_image = train_df.groupby('ImageId')['ClassId'].count()
labels_per_image.value_counts()
```

Out[0]:

```
1    6239
2     425
3       2
Name: ClassId, dtype: int64
```

In [0]:

```
my_colors = 'rgbkymc'
labels_per_image.value_counts().plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Number of class labels per image')
plt.grid()
plt.show()
```



Observations:

- There are 6239 steel sheet images possessing only one class of defect i.e. either 1,2,3 or 4
- There are 425 steel sheet images possessing a combination of 2 classes defects
- There are 2 steel sheet images possessing a combination of 3 classes defects
- There are no steel sheet images possessing a combination of 4 classes defects

3.4.3 Displaying few sample images belonging to each of the 4 classes

In [0]:

```
d1= train_df[train_df.ClassId==1]
d2= train_df[train_df.ClassId==2]
d3= train_df[train_df.ClassId==3]
d4= train_df[train_df.ClassId==4]
```

In [0]:

```
d1.head()
```

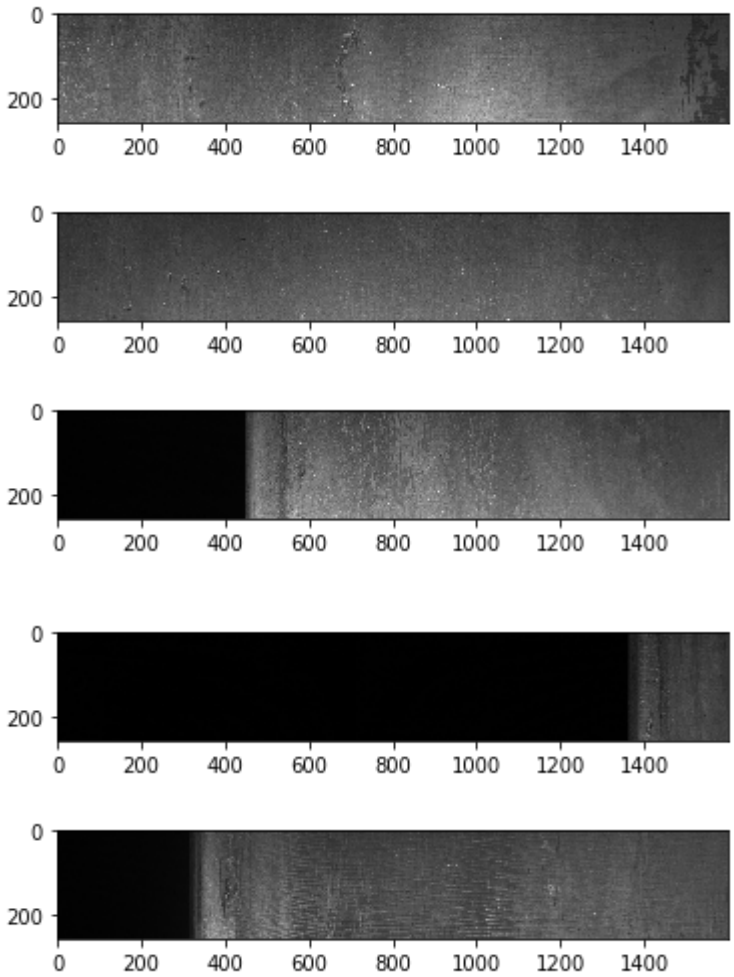
Out[0]:

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
8	002fc4e19.jpg	1	146021 3 146275 10 146529 40 146783 46 147038 ...
18	008ef3d74.jpg	1	356336 4 356587 11 356838 18 357089 25 357340 ...
21	00ac8372f.jpg	1	101742 3 101998 12 102253 19 102301 22 102509 ...

Class-1

In [0]:

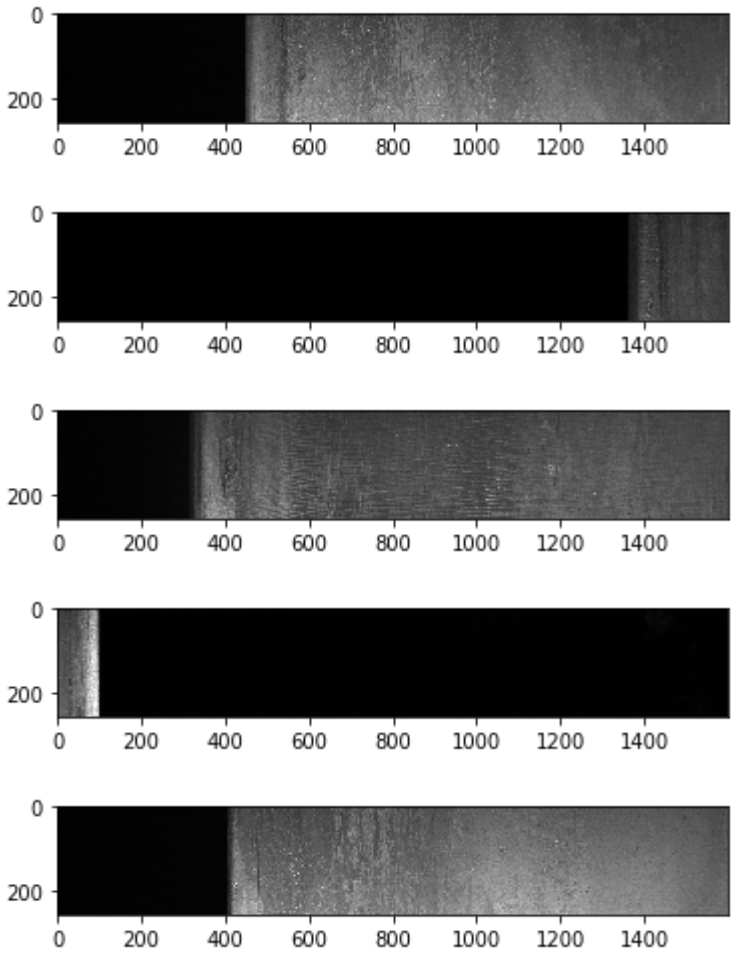
```
# load all images in a directory

from matplotlib import image
loaded_images = list()
img_names= list(d1.ImageId.values)
for filename in listdir('train_images'):
    if str(filename) in img_names:
# load image
        img_data = image.imread('train_images/' + filename)
# store loaded image
        loaded_images.append(img_data)

#displaying images
from matplotlib import pyplot
for i in range(5):
    disp= loaded_images[i]
    pyplot.imshow(disp)
    pyplot.show()
```



Class-2

In [0]:

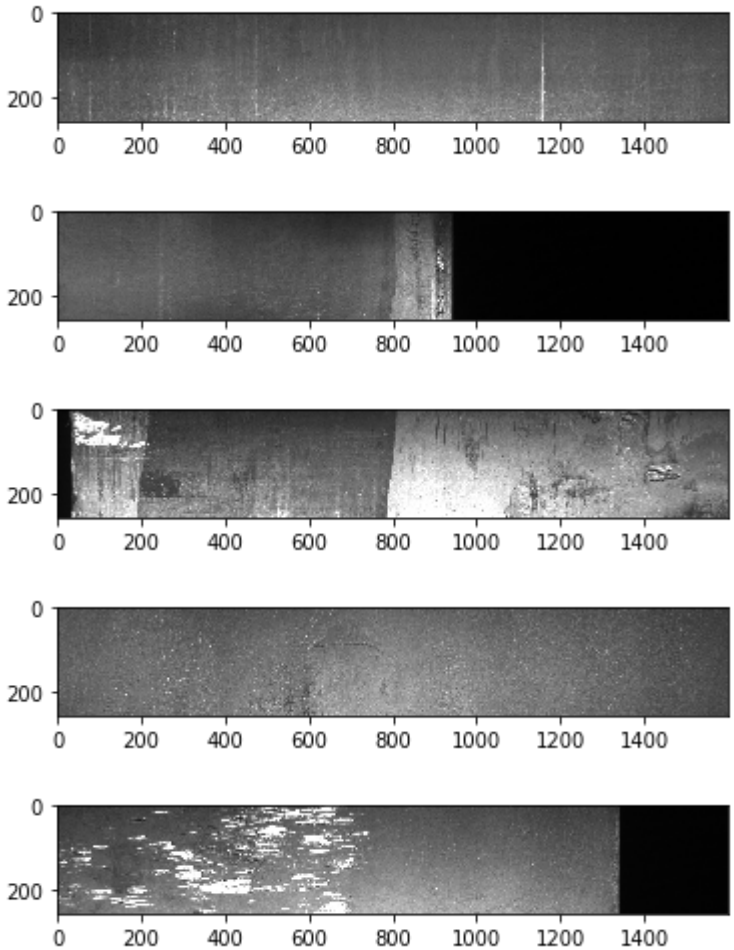
```
# load all images in a directory
loaded_images = list()
img_names= list(d2.ImageId.values)
for filename in listdir('train_images'):
    if str(filename) in img_names:
# load image
        img_data = image.imread('train_images/' + filename)
# store loaded image
        loaded_images.append(img_data)

#displaying images
from matplotlib import pyplot
for i in range(5):
    disp= loaded_images[i]
    pyplot.imshow(disp)
    pyplot.show()
```



Class-3

In [0]:

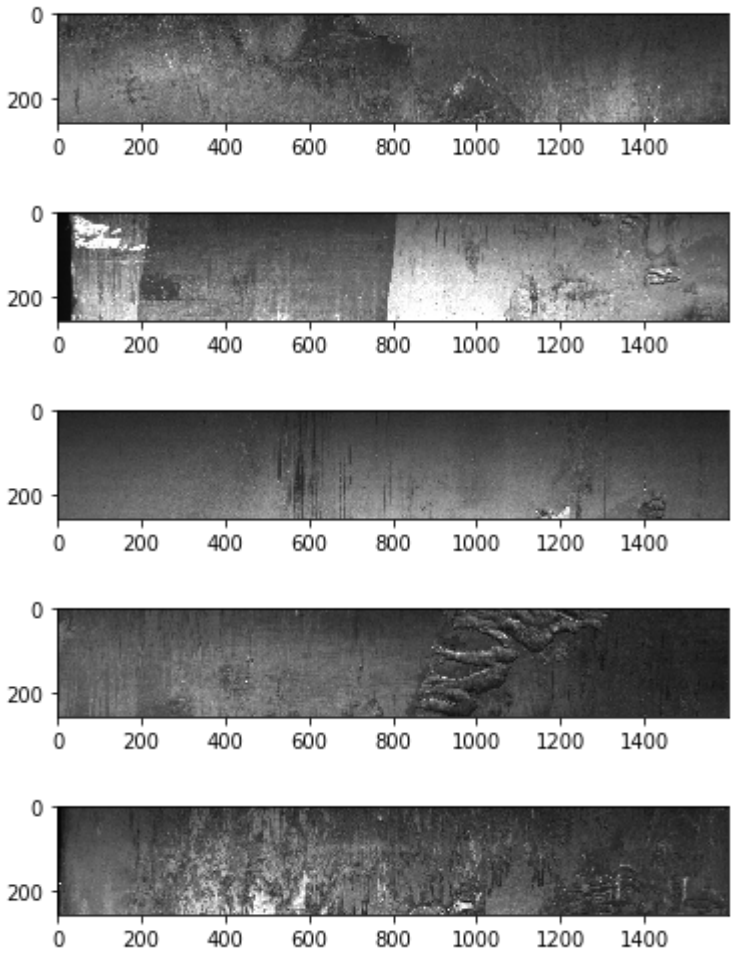
```
# load all images in a directory
loaded_images = list()
img_names= list(d3.ImageId.values)
for filename in listdir('train_images'):
    if str(filename) in img_names:
# load image
        img_data = image.imread('train_images/' + filename)
# store loaded image
        loaded_images.append(img_data)

#displaying images
from matplotlib import pyplot
for i in range(5):
    disp= loaded_images[i]
    pyplot.imshow(disp)
    pyplot.show()
```



Class-4

In [0]:

```
# load all images in a directory
loaded_images = list()
img_names= list(d4.ImageId.values)
for filename in listdir('train_images'):
    if str(filename) in img_names:
# load image
        img_data = image.imread('train_images/' + filename)
# store loaded image
        loaded_images.append(img_data)

#displaying images
from matplotlib import pyplot
for i in range(5):
    disp= loaded_images[i]
    pyplot.imshow(disp)
    pyplot.show()
```



4.0 Data preparation

4.1 Preparing defect data for each image

In [0]:

```
# preparing a dataframe with each image ID being represented with 4 classes
images= []
class_id= []
for img in listdir('train_images'):
    images.append(img)
    class_id.append(1)
    images.append(img)
    class_id.append(2)
    images.append(img)
    class_id.append(3)
    images.append(img)
    class_id.append(4)
train_images= pd.DataFrame(images,columns=['ImageId'])
train_images['ClassId'] = class_id
train_images.head()
```

Out[0]:

	ImageId	ClassId
0	0002cc93b.jpg	1
1	0002cc93b.jpg	2
2	0002cc93b.jpg	3
3	0002cc93b.jpg	4
4	00031f466.jpg	1

In [0]:

```
train_images.shape
```

Out[0]:

```
(50272, 2)
```

In [0]:

```
# Dataframe containing RLE(run length encoded pixels) representing defects
train_df.head()
```

Out[0]:

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
3	000f6bf48.jpg	4	131973 1 132228 4 132483 6 132738 8 132993 11 ...
4	0014fce06.jpg	3	229501 11 229741 33 229981 55 230221 77 230468...

In [0]:

```
# merging defect & non-defect data
temp_df = pd.merge(train_images, train_df,how='outer',on=['ImageId','ClassId'])
temp_df = temp_df.fillna('')
print(temp_df.shape)
temp_df.head()
```

```
(50272, 3)
```

Out[0]:

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0002cc93b.jpg	2	
2	0002cc93b.jpg	3	
3	0002cc93b.jpg	4	
4	00031f466.jpg	1	

In [0]:

```
# Grouping the data according to ImageID with each row representing images with single
 or multiple defects
#https://www.geeksforgeeks.org/python-pandas-pivot_table/

defect_data = pd.pivot_table(temp_df, values='EncodedPixels', index='ImageId',columns=
'ClassId', aggfunc=np.sum).astype(str)
defect_data = defect_data.reset_index()
defect_data.columns = ['ImageId','Defect_1','Defect_2','Defect_3','Defect_4']
defect_data.head()
```

Out[0]:

	ImageId	Defect_1	Defect_2	Defect_3	Defect_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 19347 110 19603 11...	

In [0]:

```
defect_data.shape
```

Out[0]:

```
(12568, 5)
```

4.2 Splitting the data into train & CV

In [0]:

```
from sklearn.model_selection import train_test_split
data= defect_data
train_data, cv_data = train_test_split(data, test_size=0.15)
print(train_data.shape)
print(cv_data.shape)
```

```
(10682, 5)
(1886, 5)
```

5.0 Utility functions

5.1 Mask encoding and decoding

In [0]:

```
# to convert masks to run length encoded values
#https://www.kaggle.com/aleksandradeis/steel-defect-detection-eda

def mask2rle(img):
    '''
    img: numpy array, 1 - mask, 0 - background
    Returns run length as string formated
    '''
    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]:

```
# to convert run length encoded pixels to masks
#https://www.kaggle.com/aleksandradeis/steel-defect-detection-eda
def rle2mask(rle):
    '''
    Returns mask array by converting run length encoded pixels

    '''
  # If rle is empty or null
    if(len(rle)<1):
        return np.zeros((128,800) ,dtype=np.uint8)

    height = 256
    width = 1600
    # Defining the length of mask. This will be a 1d array which will be reshaped to 2d
later.
    mask = np.zeros(height*width ).astype(np.uint8)
    array = np.asarray([int(x) for x in rle.split()]) # array containing rle
    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.
    for i,start in enumerate(start):
        mask[int(start):int(start+length[i])] = 1
    return mask.reshape( (height,width), order='F' )[::2,::2]
```

5.2 Custom metric and loss function

In [0]:

```
from keras import backend as K
from keras.losses import binary_crossentropy
#https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/
#https://www.kaggle.com/wh1tezzz/correct-dice-metrics-for-this-competition
#https://forums.fast.ai/t/understanding-the-dice-coefficient/5838

def dice_coef(y_true, y_pred, smooth=1):
    ''' Function that returns dice coefficient by taking input masks and predicted mask
s'''

    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): #combination of dice loss and binary cross entro
py for all pixels
    return binary_crossentropy(y_true, y_predict) + (1-dice_coef(y_true, y_predict))
```

5.3 Training Plots

In [0]:

```
from matplotlib import pyplot
def plot(history):
    '''function to plot epoch vs bce_dice_loss & epoch vs dice_coeff '''
    # plot bce_dice_loss
    pyplot.subplot(121)
    pyplot.title('bce_dice_loss')
    pyplot.xlabel('Epoch')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='CV')
    # plot dice_coeff
    pyplot.subplot(122)
    pyplot.title('Dice_coef')
    pyplot.xlabel('Epoch')
    pyplot.plot(history.history['dice_coef'], color='blue', label='train')
    pyplot.plot(history.history['val_dice_coef'], color='orange', label='CV')
```

5.4 Visualizing defects on train & validation images

In [0]:

```
def visualize_defects(data,model):
    ''' Function that takes data containing imageID's ,model and outputs 4 masks for ea
ch image '''

    import random
    image_id= list(data.ImageId.values)
    for i in random.sample(image_id, 5):
        df= data[data.ImageId==i]
        X = np.empty((1,128,800,3),dtype=np.float32)
        img = cv2.imread('/content/' +'train_images/' + i)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (800,128))
        X[0,] = img
        mask_pred = model.predict(X)
        fig, axs = plt.subplots(4,3,figsize=(18, 7))
        axs[0,0].imshow(img)
        axs[0,0].set_title(i)
        for j in range(4):

            if j<3:
                axs[j+1,0].axis('off')
            k=0
            gt= rle2mask(df.iloc[0][j+1])
            m = mask_pred[0,:,:,j].round().astype(int)
            axs[j,k+1].imshow(gt)
            axs[j,k+1].set_title('Ground truth mask: Class_'+str(j+1))
            axs[j,k+2].imshow(m)
            axs[j,k+2].set_title('Predicted mask: Class_'+str(j+1))
        plt.show()
        print('-'*100)
```

5.5 Visualizing defects on raw test images

In [0]:

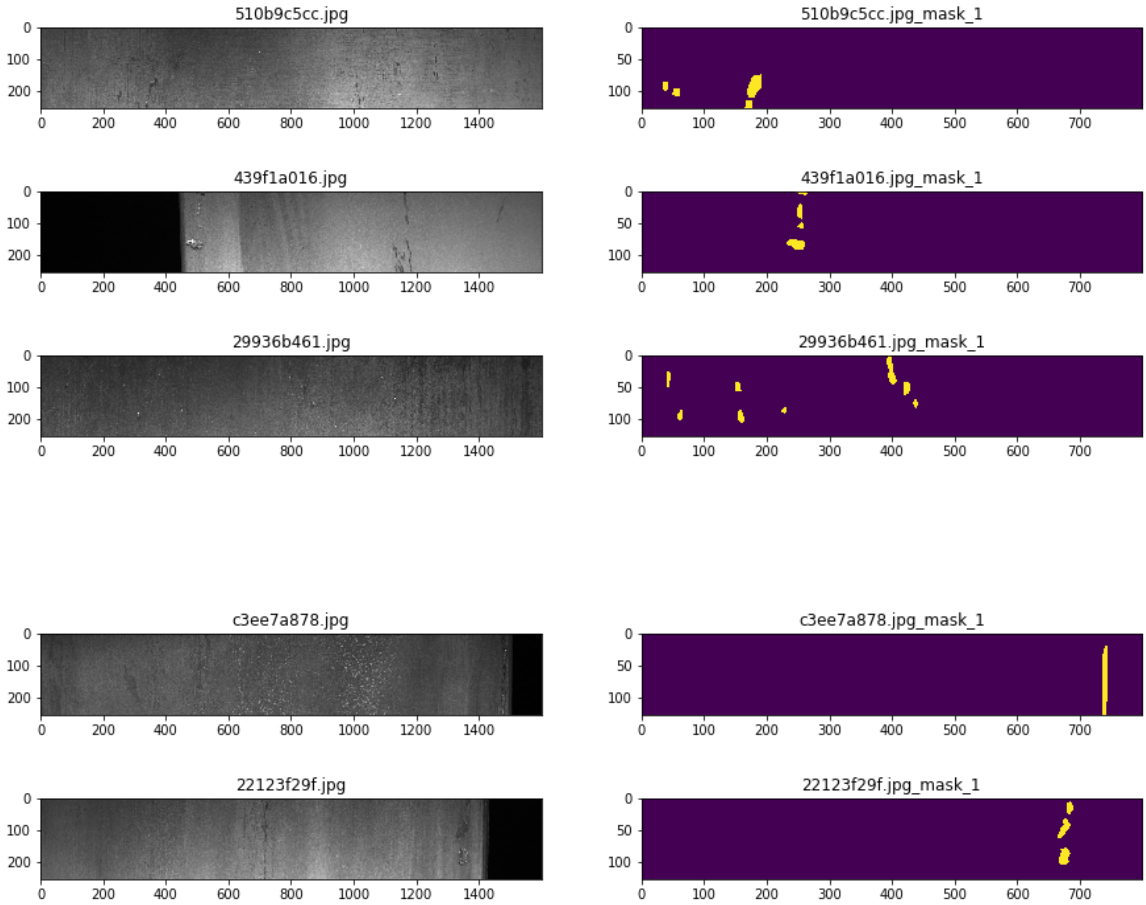
```
def visualize_defects_test(data,n):

    ''' Function that takes test ImageID's and the number of images to display as input
and therefore outputs
    mask with class ID'''

    data= data[data['EncodedPixels']!='']
    for i in random.sample(list(data['ImageId_ClassId'].values), n):
        df= data[data.ImageId_ClassId==i]
        img = Image.open('/content/' +'test_images/' + i.split('_')[0])
        pixels= df.iloc[0][1]
        mask= rle2mask(pixels)
        fig, (ax1,ax2) = plt.subplots(1,2,figsize=(15, 7))
        ax1.imshow(img)
        ax1.set_title(i.split('_')[0])
        ax2.imshow(mask)
        ax2.set_title('Predicted mask with Class-'+i.split('_')[1]+' defect.')
        plt.show()
        print('-'*150)
```

6.0 Check rle2mask function and visualize few masks & images

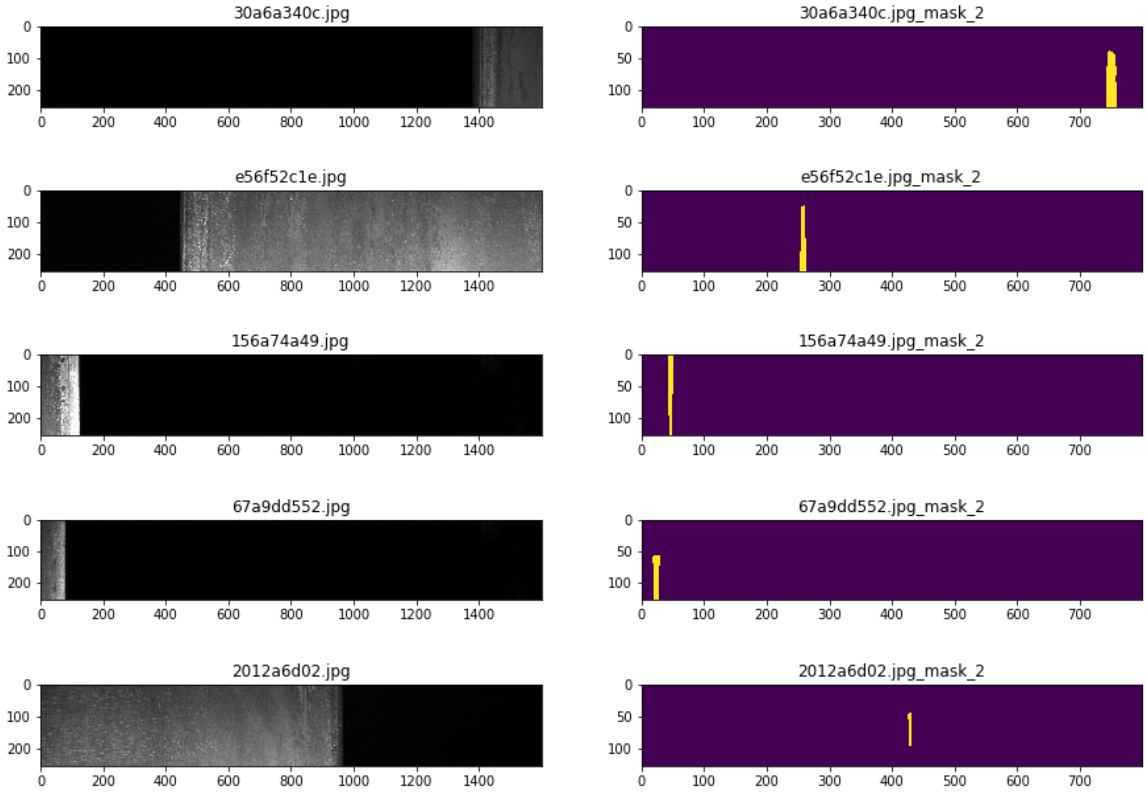
In [0]:

```
for k in [1,2,3,4]: #classes
    tmp = []
    cnt=0
    print("Sample images with Class {} defect:".format(k))
    for i in train_data[train_data[f'Defect_{k}']!=''][['ImageId',f'Defect_{k}']].value
s:
        if cnt<5:
            fig, (ax1,ax2) = plt.subplots(nrows = 1,ncols = 2,figsize=(15, 7))
            img = Image.open('train_images/' + str(i[0]))
            ax1.imshow(img)
            ax1.set_title(i[0])
            cnt+=1
            ax2.imshow(rle2mask(i[1]))
            ax2.set_title(i[0]+'_mask_'+str(k))
            plt.show()
    print('-'*100)
```

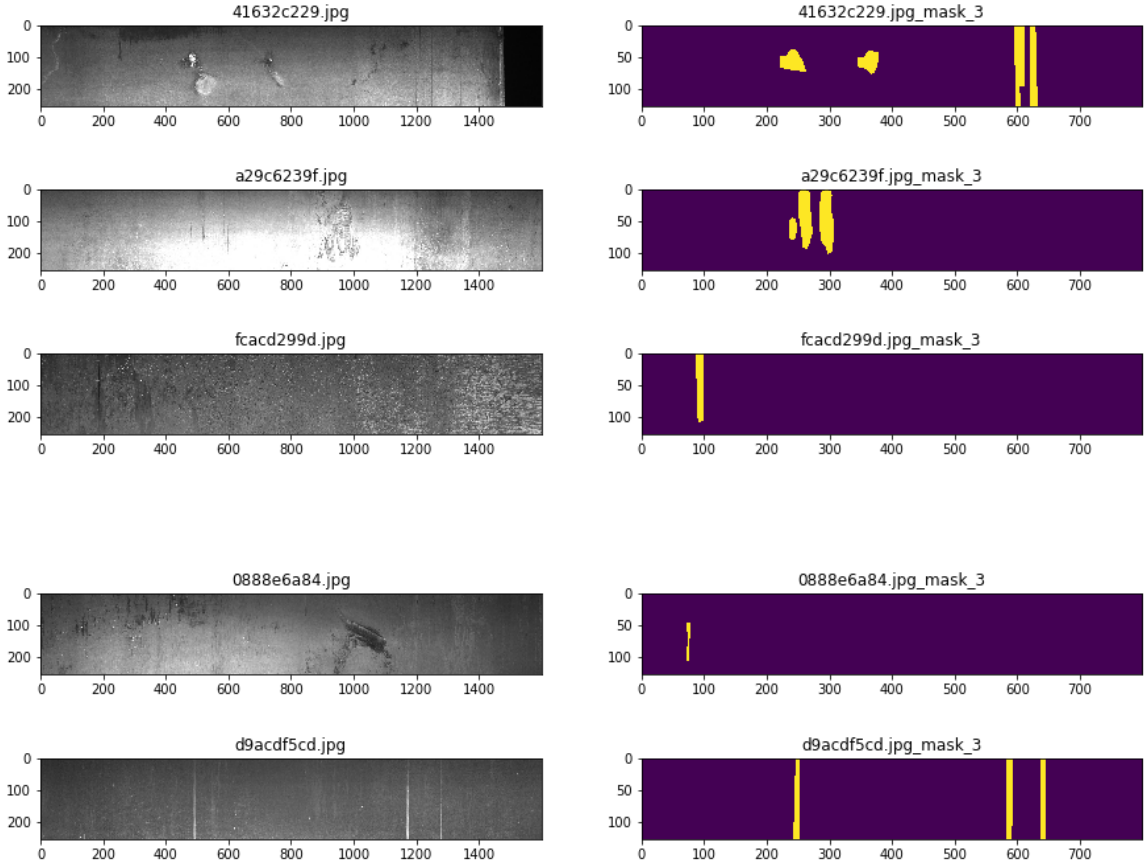
Sample images with Class 1 defect:



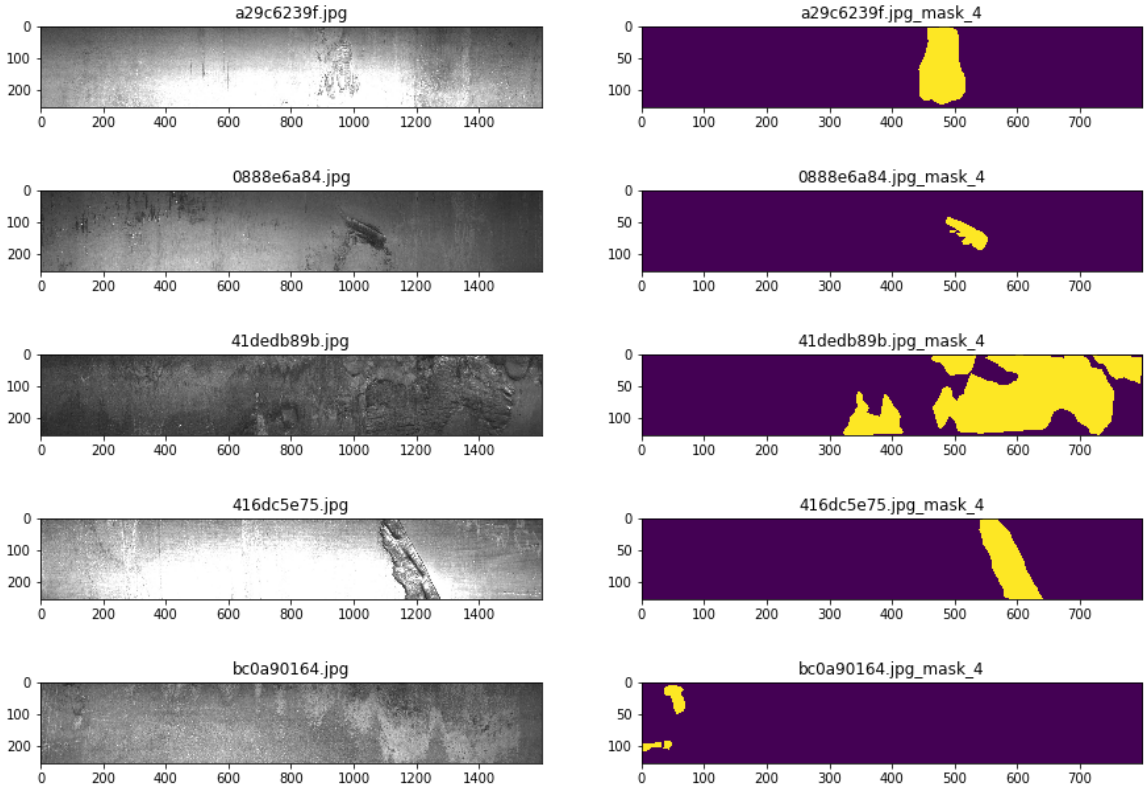
Sample images with Class 2 defect:



Sample images with Class 3 defect:



Sample images with Class 4 defect:



7.0 Generating data for Keras model

7.1 Train generator

In [0]:

```
# https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly
# https://keras.io/preprocessing/image/

import keras
from keras.preprocessing.image import ImageDataGenerator
class Train_DataGenerator(keras.utils.Sequence):
    def __init__(self, df, batch_size = 16,shuffle=False,
                 preprocess=None, info={}):
        super().__init__()
        self.df = df
        self.shuffle = shuffle
        self.batch_size = batch_size
        self.preprocess = preprocess
        self.info = info
        self.data_path = '/content/' + '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, 'flip_vertical' : True}
        ''''
        Performing data augmentation on images and the masks generated which includes H
orizontal flip &  Vertical flip

        '''
        X = np.empty((self.batch_size,128,800,3),dtype=np.float32) #images
        y = np.empty((self.batch_size,128,800,4),dtype=np.int8)    #masks
        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,] = train_datagen.apply_transform(x = img, transform_parameters = param
)
                #run-length encoding on the pixel values
            for j in range(4):
                mask= rle2mask(self.df['Defect_'+str(j+1)].iloc[indexes[i]])
                y[i,:,:,j] = train_datagen.apply_transform(x = mask, transform_paramete
rs = param)
        if self.preprocess!=None: X = self.preprocess(X)
        return X, y
```

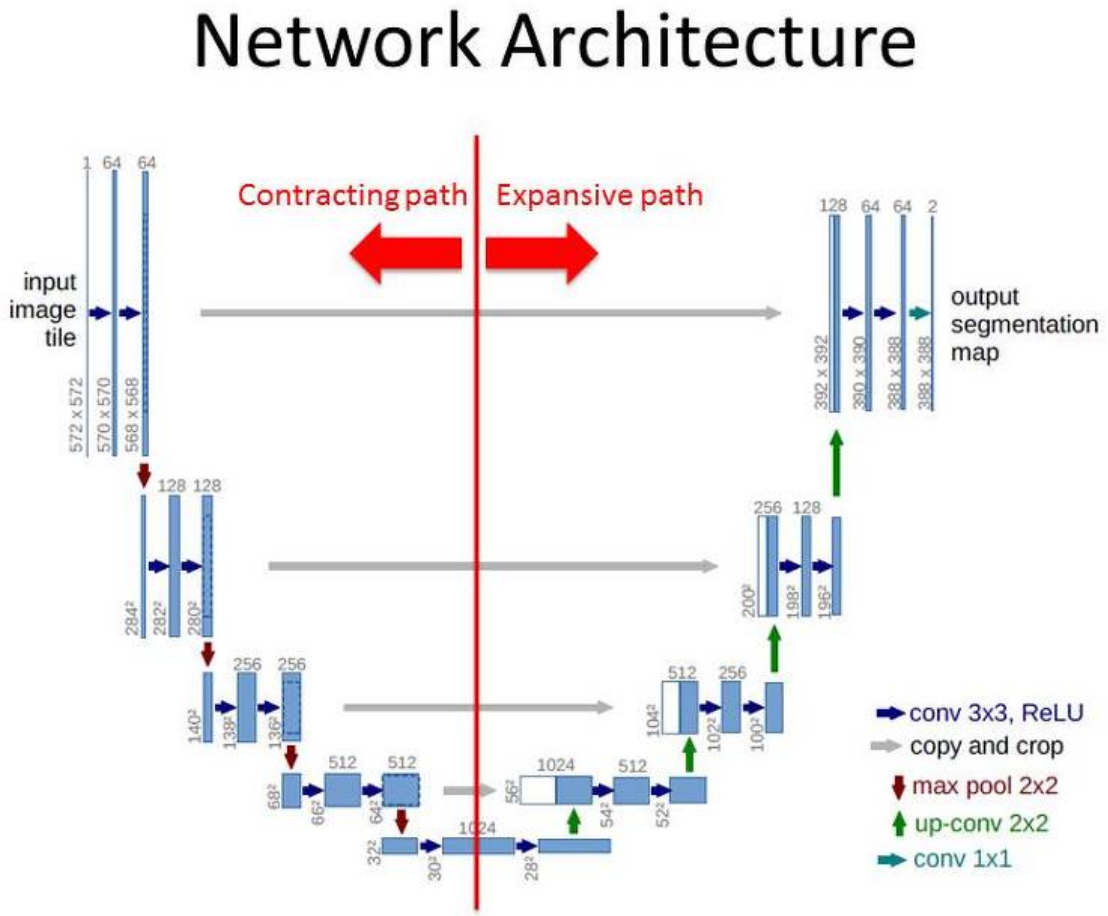
7.2 Validation generator(Without Augmentation)

In [0]:

```
# https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

import keras
from keras.preprocessing.image import ImageDataGenerator
class Val_DataGenerator(keras.utils.Sequence):
    def __init__(self, df, batch_size = 16,shuffle=False,
                 preprocess=None, info={}):
        super().__init__()
        self.df = df
        self.shuffle = shuffle
        self.batch_size = batch_size
        self.preprocess = preprocess
        self.info = info
        self.data_path = '/content/' + '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) #images
        y = np.empty((self.batch_size,128,800,4),dtype=np.int8)    #masks
        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
            X[i,] = Image.open(self.data_path + f).resize((800,128))
            #run-length encoding on the pixel values
            for j in range(4):
                y[i,:,:,j] = rle2mask(self.df['Defect_'+str(j+1)].iloc[indexes[i]])
        if self.preprocess!=None: X = self.preprocess(X)
        return X, y
```

Implementing Segmentation Architectures

1.0 Basic U-NET (First-cut approach)



References-

- https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47 (https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47)

In [0]:

```
def conv2d_block(input_tensor, n_filters, kernel_size = 3, batchnorm = True):
    """Function to add 2 convolutional layers with the parameters passed to it"""
    # first layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
              kernel_initializer = 'he_normal', padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # second layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
              kernel_initializer = 'he_normal', padding = 'same')(x)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    return x
```

In [0]:

```
def get_unet(input_img, n_filters, dropout, batchnorm):

    """Function to define the UNET architecture"""

    # Contracting Path
    c1 = conv2d_block(input_img, n_filters * 1, kernel_size = 3, batchnorm = batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)

    c2 = conv2d_block(p1, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)

    c3 = conv2d_block(p2, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)
    p3 = MaxPooling2D((2, 2))(c3)
    p3 = Dropout(dropout)(p3)

    c4 = conv2d_block(p3, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    p4 = MaxPooling2D((2, 2))(c4)
    p4 = Dropout(dropout)(p4)

    c5 = conv2d_block(p4, n_filters = n_filters * 16, kernel_size = 3, batchnorm = batc
hnorm)

    # Expansive Path
    u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides = (2, 2), padding = 'same')(c5)
    u6 = concatenate([u6, c4])
    u6 = Dropout(dropout)(u6)
    c6 = conv2d_block(u6, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)

    u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides = (2, 2), padding = 'same')(c6)
    u7 = concatenate([u7, c3])
    u7 = Dropout(dropout)(u7)
    c7 = conv2d_block(u7, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)

    u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides = (2, 2), padding = 'same')(c7)
    u8 = concatenate([u8, c2])
    u8 = Dropout(dropout)(u8)
    c8 = conv2d_block(u8, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)

    u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides = (2, 2), padding = 'same')(c8)
    u9 = concatenate([u9, c1])
    u9 = Dropout(dropout)(u9)
    c9 = conv2d_block(u9, n_filters * 1, kernel_size = 3, batchnorm = batchnorm)

    outputs = Conv2D(4, (1, 1), activation='sigmoid')(c9)
    model = Model(inputs=[input_img], outputs=[outputs])
    return model
```

In [0]:

```
input_img = Input((128, 800, 3), name='img')
model = get_unet(input_img, n_filters=8, dropout=0.2, batchnorm=True)
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
img (InputLayer)             (None, 128, 800, 3)       0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 128, 800, 8)       224
img[0][0]
_________________________________________________________________
batch_normalization_19 (BatchNo (None, 128, 800, 8)       32
conv2d_20[0][0]
_________________________________________________________________
activation_19 (Activation)   (None, 128, 800, 8)       0
batch_normalization_19[0][0]
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 128, 800, 8)       584
activation_19[0][0]
_________________________________________________________________
batch_normalization_20 (BatchNo (None, 128, 800, 8)       32
conv2d_21[0][0]
_________________________________________________________________
activation_20 (Activation)   (None, 128, 800, 8)       0
batch_normalization_20[0][0]
_________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 64, 400, 8)        0
activation_20[0][0]
_________________________________________________________________
dropout_9 (Dropout)          (None, 64, 400, 8)        0
max_pooling2d_5[0][0]
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 64, 400, 16)       1168
dropout_9[0][0]
_________________________________________________________________
batch_normalization_21 (BatchNo (None, 64, 400, 16)       64
conv2d_22[0][0]
_________________________________________________________________
activation_21 (Activation)   (None, 64, 400, 16)       0
batch_normalization_21[0][0]
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 64, 400, 16)       2320
activation_21[0][0]
_________________________________________________________________
batch_normalization_22 (BatchNo (None, 64, 400, 16)       64
conv2d_23[0][0]
_________________________________________________________________
```

```
activation_22 (Activation)      (None, 64, 400, 16)  0           batch_nor
malization_22[0][0]
__________________________________________________________________________
_____________________
max_pooling2d_6 (MaxPooling2D)  (None, 32, 200, 16)  0           activatio
n_22[0][0]
__________________________________________________________________________
_____________________
dropout_10 (Dropout)            (None, 32, 200, 16)  0           max_pooli
ng2d_6[0][0]
__________________________________________________________________________
_____________________
conv2d_24 (Conv2D)              (None, 32, 200, 32)  4640        dropout_1
0[0][0]
__________________________________________________________________________
_____________________
batch_normalization_23 (BatchNo (None, 32, 200, 32)  128         conv2d_24
[0][0]
__________________________________________________________________________
_____________________
activation_23 (Activation)      (None, 32, 200, 32)  0           batch_nor
malization_23[0][0]
__________________________________________________________________________
_____________________
conv2d_25 (Conv2D)              (None, 32, 200, 32)  9248        activatio
n_23[0][0]
__________________________________________________________________________
_____________________
batch_normalization_24 (BatchNo (None, 32, 200, 32)  128         conv2d_25
[0][0]
__________________________________________________________________________
_____________________
activation_24 (Activation)      (None, 32, 200, 32)  0           batch_nor
malization_24[0][0]
__________________________________________________________________________
_____________________
max_pooling2d_7 (MaxPooling2D)  (None, 16, 100, 32)  0           activatio
n_24[0][0]
__________________________________________________________________________
_____________________
dropout_11 (Dropout)            (None, 16, 100, 32)  0           max_pooli
ng2d_7[0][0]
__________________________________________________________________________
_____________________
conv2d_26 (Conv2D)              (None, 16, 100, 64)  18496       dropout_1
1[0][0]
__________________________________________________________________________
_____________________
batch_normalization_25 (BatchNo (None, 16, 100, 64)  256         conv2d_26
[0][0]
__________________________________________________________________________
_____________________
activation_25 (Activation)      (None, 16, 100, 64)  0           batch_nor
malization_25[0][0]
__________________________________________________________________________
_____________________
conv2d_27 (Conv2D)              (None, 16, 100, 64)  36928       activatio
n_25[0][0]
__________________________________________________________________________
_____________________
batch_normalization_26 (BatchNo (None, 16, 100, 64)  256         conv2d_27
```

```
[0][0]
__________________________________________________________________________________________
_____________________
activation_26 (Activation)      (None, 16, 100, 64)  0           batch_nor
malization_26[0][0]
__________________________________________________________________________________________
_____________________
max_pooling2d_8 (MaxPooling2D)  (None, 8, 50, 64)    0           activatio
n_26[0][0]
__________________________________________________________________________________________
_____________________
dropout_12 (Dropout)            (None, 8, 50, 64)    0           max_pooli
ng2d_8[0][0]
__________________________________________________________________________________________
_____________________
conv2d_28 (Conv2D)              (None, 8, 50, 128)   73856       dropout_1
2[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_27 (BatchNo (None, 8, 50, 128)   512         conv2d_28
[0][0]
__________________________________________________________________________________________
_____________________
activation_27 (Activation)      (None, 8, 50, 128)   0           batch_nor
malization_27[0][0]
__________________________________________________________________________________________
_____________________
conv2d_29 (Conv2D)              (None, 8, 50, 128)   147584      activatio
n_27[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_28 (BatchNo (None, 8, 50, 128)   512         conv2d_29
[0][0]
__________________________________________________________________________________________
_____________________
activation_28 (Activation)      (None, 8, 50, 128)   0           batch_nor
malization_28[0][0]
__________________________________________________________________________________________
_____________________
conv2d_transpose_5 (Conv2DTrans (None, 16, 100, 64)  73792       activatio
n_28[0][0]
__________________________________________________________________________________________
_____________________
concatenate_5 (Concatenate)     (None, 16, 100, 128) 0           conv2d_tr
anspose_5[0][0]
                                                                 activatio
n_26[0][0]
__________________________________________________________________________________________
_____________________
dropout_13 (Dropout)            (None, 16, 100, 128) 0           concatena
te_5[0][0]
__________________________________________________________________________________________
_____________________
conv2d_30 (Conv2D)              (None, 16, 100, 64)  73792       dropout_1
3[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_29 (BatchNo (None, 16, 100, 64)  256         conv2d_30
[0][0]
__________________________________________________________________________________________
_____________________
```

```
activation_29 (Activation)      (None, 16, 100, 64)  0           batch_nor
malization_29[0][0]
__________________________________________________________________________
_____________________
conv2d_31 (Conv2D)              (None, 16, 100, 64)  36928       activatio
n_29[0][0]
__________________________________________________________________________
_____________________
batch_normalization_30 (BatchNo (None, 16, 100, 64)  256         conv2d_31
[0][0]
__________________________________________________________________________
_____________________
activation_30 (Activation)      (None, 16, 100, 64)  0           batch_nor
malization_30[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_6 (Conv2DTrans (None, 32, 200, 32)  18464       activatio
n_30[0][0]
__________________________________________________________________________
_____________________
concatenate_6 (Concatenate)     (None, 32, 200, 64)  0           conv2d_tr
anspose_6[0][0]
                                                                 activatio
n_24[0][0]
__________________________________________________________________________
_____________________
dropout_14 (Dropout)            (None, 32, 200, 64)  0           concatena
te_6[0][0]
__________________________________________________________________________
_____________________
conv2d_32 (Conv2D)              (None, 32, 200, 32)  18464       dropout_1
4[0][0]
__________________________________________________________________________
_____________________
batch_normalization_31 (BatchNo (None, 32, 200, 32)  128         conv2d_32
[0][0]
__________________________________________________________________________
_____________________
activation_31 (Activation)      (None, 32, 200, 32)  0           batch_nor
malization_31[0][0]
__________________________________________________________________________
_____________________
conv2d_33 (Conv2D)              (None, 32, 200, 32)  9248        activatio
n_31[0][0]
__________________________________________________________________________
_____________________
batch_normalization_32 (BatchNo (None, 32, 200, 32)  128         conv2d_33
[0][0]
__________________________________________________________________________
_____________________
activation_32 (Activation)      (None, 32, 200, 32)  0           batch_nor
malization_32[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_7 (Conv2DTrans (None, 64, 400, 16)  4624        activatio
n_32[0][0]
__________________________________________________________________________
_____________________
concatenate_7 (Concatenate)     (None, 64, 400, 32)  0           conv2d_tr
anspose_7[0][0]
                                                                 activatio
```

```
n_22[0][0]
__________________________________________________________________________________________________
__________________________
dropout_15 (Dropout)            (None, 64, 400, 32)  0           concatena
te_7[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_34 (Conv2D)              (None, 64, 400, 16)  4624        dropout_1
5[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_33 (BatchNo (None, 64, 400, 16)  64          conv2d_34
[0][0]
__________________________________________________________________________________________________
__________________________
activation_33 (Activation)      (None, 64, 400, 16)  0           batch_nor
malization_33[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_35 (Conv2D)              (None, 64, 400, 16)  2320        activatio
n_33[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_34 (BatchNo (None, 64, 400, 16)  64          conv2d_35
[0][0]
__________________________________________________________________________________________________
__________________________
activation_34 (Activation)      (None, 64, 400, 16)  0           batch_nor
malization_34[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_transpose_8 (Conv2DTrans (None, 128, 800, 8)  1160        activatio
n_34[0][0]
__________________________________________________________________________________________________
__________________________
concatenate_8 (Concatenate)     (None, 128, 800, 16) 0           conv2d_tr
anspose_8[0][0]
                                                                 activatio
n_20[0][0]
__________________________________________________________________________________________________
__________________________
dropout_16 (Dropout)            (None, 128, 800, 16) 0           concatena
te_8[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_36 (Conv2D)              (None, 128, 800, 8)  1160        dropout_1
6[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_35 (BatchNo (None, 128, 800, 8)  32          conv2d_36
[0][0]
__________________________________________________________________________________________________
__________________________
activation_35 (Activation)      (None, 128, 800, 8)  0           batch_nor
malization_35[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_37 (Conv2D)              (None, 128, 800, 8)  584         activatio
n_35[0][0]
__________________________________________________________________________________________________
__________________________
```

```
batch_normalization_36 (BatchNo (None, 128, 800, 8)  32          conv2d_37
[0][0]
__________________________________________________________________________
_____________________
activation_36 (Activation)      (None, 128, 800, 8)  0           batch_nor
malization_36[0][0]
__________________________________________________________________________
_____________________
conv2d_38 (Conv2D)              (None, 128, 800, 4)  36          activatio
n_36[0][0]
==========================================================================
=======================
Total params: 543,188
Trainable params: 541,716
Non-trainable params: 1,472
__________________________________________________________________________
_____________________
```

1.1 Checkpointing the model and creating the callback list

In [0]:

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import CSVLogger
import matplotlib.pyplot as plt
from tensorflow.python.keras.callbacks import TensorBoard
from keras.callbacks import TensorBoard
import tensorflow as tf
import datetime
import keras
from tensorboardcolab import *
from keras.callbacks import ReduceLROnPlateau

#https://github.com/taomanwai/tensorboardcolab/
tbc=TensorBoardColab()
#https://machinelearningmastery.com/check-point-deep-learning-models-keras/
mc = ModelCheckpoint('best_model_unet.h5', monitor='val_dice_coef', verbose=1, save_bes
t_only=True, mode='max')
callbacks_list = [mc, TensorBoardColabCallback(tbc)]
```

```
Wait for 8 seconds...
TensorBoard link:
https://bde24643.ngrok.io
```

1.2 Fitting the train data and validation

In [0]:

```
train_batches = Train_DataGenerator(train_data,shuffle=True)
valid_batches = Val_DataGenerator(cv_data)
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs =
50, verbose=1,
                              class_weight= class_wts,callbacks= callbacks_list) #class
_wts as clculated in EDA
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Pleas
e use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please us
e tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/core.py:49: The name tf.summary.FileWriter is deprecated. Please use
tf.compat.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callb
acks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.c
ompat.v1.summary.merge_all instead.

Epoch 1/50
667/667 [==============================] - 235s 352ms/step - loss: 1.2888
- dice_coef: 0.0286 - val_loss: 0.9921 - val_dice_coef: 0.0688

Epoch 00001: val_dice_coef improved from -inf to 0.06879, saving model to
/content/weights.best.hdf5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.co
mpat.v1.Summary instead.

Epoch 2/50
667/667 [==============================] - 221s 332ms/step - loss: 0.7664
- dice_coef: 0.2798 - val_loss: 0.6661 - val_dice_coef: 0.3743

Epoch 00002: val_dice_coef improved from 0.06879 to 0.37426, saving model
to /content/weights.best.hdf5
Epoch 3/50
667/667 [==============================] - 218s 327ms/step - loss: 0.6021
- dice_coef: 0.4394 - val_loss: 0.6355 - val_dice_coef: 0.4068

Epoch 00003: val_dice_coef improved from 0.37426 to 0.40684, saving model
to /content/weights.best.hdf5
Epoch 4/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5688
- dice_coef: 0.4727 - val_loss: 0.5948 - val_dice_coef: 0.4483

Epoch 00004: val_dice_coef improved from 0.40684 to 0.44832, saving model
to /content/weights.best.hdf5
Epoch 5/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5578
- dice_coef: 0.4831 - val_loss: 0.5886 - val_dice_coef: 0.4532

Epoch 00005: val_dice_coef improved from 0.44832 to 0.45317, saving model
to /content/weights.best.hdf5
Epoch 6/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5479
- dice_coef: 0.4926 - val_loss: 0.6220 - val_dice_coef: 0.4202

Epoch 00006: val_dice_coef did not improve from 0.45317
Epoch 7/50
667/667 [==============================] - 219s 328ms/step - loss: 0.5365
- dice_coef: 0.5031 - val_loss: 0.5678 - val_dice_coef: 0.4741

Epoch 00007: val_dice_coef improved from 0.45317 to 0.47414, saving model
to /content/weights.best.hdf5
```

```
Epoch 8/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5266
- dice_coef: 0.5121 - val_loss: 0.5805 - val_dice_coef: 0.4598

Epoch 00008: val_dice_coef did not improve from 0.47414
Epoch 9/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5284
- dice_coef: 0.5105 - val_loss: 0.6018 - val_dice_coef: 0.4460

Epoch 00009: val_dice_coef did not improve from 0.47414
Epoch 10/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5184
- dice_coef: 0.5197 - val_loss: 0.6106 - val_dice_coef: 0.4335

Epoch 00010: val_dice_coef did not improve from 0.47414
Epoch 11/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5083
- dice_coef: 0.5291 - val_loss: 0.5716 - val_dice_coef: 0.4732

Epoch 00011: val_dice_coef did not improve from 0.47414
Epoch 12/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5064
- dice_coef: 0.5309 - val_loss: 0.5848 - val_dice_coef: 0.4641

Epoch 00012: val_dice_coef did not improve from 0.47414
Epoch 13/50
667/667 [==============================] - 218s 328ms/step - loss: 0.5143
- dice_coef: 0.5233 - val_loss: 0.5851 - val_dice_coef: 0.4597

Epoch 00013: val_dice_coef did not improve from 0.47414
Epoch 14/50
667/667 [==============================] - 218s 327ms/step - loss: 0.5044
- dice_coef: 0.5327 - val_loss: 0.5585 - val_dice_coef: 0.4843

Epoch 00014: val_dice_coef improved from 0.47414 to 0.48428, saving model
to /content/weights.best.hdf5
Epoch 15/50
667/667 [==============================] - 219s 329ms/step - loss: 0.4969
- dice_coef: 0.5393 - val_loss: 0.5834 - val_dice_coef: 0.4602

Epoch 00015: val_dice_coef did not improve from 0.48428
Epoch 16/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4981
- dice_coef: 0.5378 - val_loss: 0.5541 - val_dice_coef: 0.4882

Epoch 00016: val_dice_coef improved from 0.48428 to 0.48824, saving model
to /content/weights.best.hdf5
Epoch 17/50
667/667 [==============================] - 220s 329ms/step - loss: 0.4986
- dice_coef: 0.5371 - val_loss: 0.5414 - val_dice_coef: 0.4994

Epoch 00017: val_dice_coef improved from 0.48824 to 0.49937, saving model
to /content/weights.best.hdf5
Epoch 18/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4927
- dice_coef: 0.5427 - val_loss: 0.5506 - val_dice_coef: 0.4906

Epoch 00018: val_dice_coef did not improve from 0.49937
Epoch 19/50
667/667 [==============================] - 220s 329ms/step - loss: 0.4946
- dice_coef: 0.5407 - val_loss: 0.5521 - val_dice_coef: 0.4894
```

```
Epoch 00019: val_dice_coef did not improve from 0.49937
Epoch 20/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4869
- dice_coef: 0.5477 - val_loss: 0.5352 - val_dice_coef: 0.5063

Epoch 00020: val_dice_coef improved from 0.49937 to 0.50629, saving model
to /content/weights.best.hdf5
Epoch 21/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4799
- dice_coef: 0.5545 - val_loss: 0.5675 - val_dice_coef: 0.4741

Epoch 00021: val_dice_coef did not improve from 0.50629
Epoch 22/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4698
- dice_coef: 0.5637 - val_loss: 0.5634 - val_dice_coef: 0.4798

Epoch 00022: val_dice_coef did not improve from 0.50629
Epoch 23/50
667/667 [==============================] - 219s 328ms/step - loss: 0.4649
- dice_coef: 0.5683 - val_loss: 0.5331 - val_dice_coef: 0.5066

Epoch 00023: val_dice_coef improved from 0.50629 to 0.50659, saving model
to /content/weights.best.hdf5
Epoch 24/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4522
- dice_coef: 0.5803 - val_loss: 0.5130 - val_dice_coef: 0.5247

Epoch 00024: val_dice_coef improved from 0.50659 to 0.52471, saving model
to /content/weights.best.hdf5
Epoch 25/50
667/667 [==============================] - 220s 329ms/step - loss: 0.4521
- dice_coef: 0.5806 - val_loss: 0.4926 - val_dice_coef: 0.5441

Epoch 00025: val_dice_coef improved from 0.52471 to 0.54411, saving model
to /content/weights.best.hdf5
Epoch 26/50
667/667 [==============================] - 222s 332ms/step - loss: 0.4461
- dice_coef: 0.5862 - val_loss: 0.4957 - val_dice_coef: 0.5411

Epoch 00026: val_dice_coef did not improve from 0.54411
Epoch 27/50
667/667 [==============================] - 224s 335ms/step - loss: 0.4415
- dice_coef: 0.5903 - val_loss: 0.5474 - val_dice_coef: 0.4974

Epoch 00027: val_dice_coef did not improve from 0.54411
Epoch 28/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4380
- dice_coef: 0.5938 - val_loss: 0.5168 - val_dice_coef: 0.5255

Epoch 00028: val_dice_coef did not improve from 0.54411
Epoch 29/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4381
- dice_coef: 0.5933 - val_loss: 0.4751 - val_dice_coef: 0.5607

Epoch 00029: val_dice_coef improved from 0.54411 to 0.56069, saving model
to /content/weights.best.hdf5
Epoch 30/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4303
- dice_coef: 0.6009 - val_loss: 0.4815 - val_dice_coef: 0.5556
```

```
Epoch 00030: val_dice_coef did not improve from 0.56069
Epoch 31/50
667/667 [==============================] - 221s 332ms/step - loss: 0.4316
- dice_coef: 0.5998 - val_loss: 0.4972 - val_dice_coef: 0.5385

Epoch 00031: val_dice_coef did not improve from 0.56069
Epoch 32/50
667/667 [==============================] - 220s 330ms/step - loss: 0.4255
- dice_coef: 0.6053 - val_loss: 0.4764 - val_dice_coef: 0.5602

Epoch 00032: val_dice_coef did not improve from 0.56069
Epoch 33/50
667/667 [==============================] - 222s 332ms/step - loss: 0.4228
- dice_coef: 0.6079 - val_loss: 0.4917 - val_dice_coef: 0.5472

Epoch 00033: val_dice_coef did not improve from 0.56069
Epoch 34/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4186
- dice_coef: 0.6119 - val_loss: 0.4691 - val_dice_coef: 0.5668

Epoch 00034: val_dice_coef improved from 0.56069 to 0.56676, saving model
to /content/weights.best.hdf5
Epoch 35/50
667/667 [==============================] - 223s 334ms/step - loss: 0.4233
- dice_coef: 0.6072 - val_loss: 0.4857 - val_dice_coef: 0.5514

Epoch 00035: val_dice_coef did not improve from 0.56676
Epoch 36/50
667/667 [==============================] - 223s 335ms/step - loss: 0.4080
- dice_coef: 0.6219 - val_loss: 0.4777 - val_dice_coef: 0.5587

Epoch 00036: val_dice_coef did not improve from 0.56676
Epoch 37/50
667/667 [==============================] - 221s 332ms/step - loss: 0.4142
- dice_coef: 0.6159 - val_loss: 0.4701 - val_dice_coef: 0.5660

Epoch 00037: val_dice_coef did not improve from 0.56676
Epoch 38/50
667/667 [==============================] - 221s 332ms/step - loss: 0.4136
- dice_coef: 0.6163 - val_loss: 0.4693 - val_dice_coef: 0.5670

Epoch 00038: val_dice_coef improved from 0.56676 to 0.56701, saving model
to /content/weights.best.hdf5
Epoch 39/50
667/667 [==============================] - 221s 332ms/step - loss: 0.4096
- dice_coef: 0.6203 - val_loss: 0.4714 - val_dice_coef: 0.5647

Epoch 00039: val_dice_coef did not improve from 0.56701
Epoch 40/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4091
- dice_coef: 0.6204 - val_loss: 0.4560 - val_dice_coef: 0.5791

Epoch 00040: val_dice_coef improved from 0.56701 to 0.57912, saving model
to /content/weights.best.hdf5
Epoch 41/50
667/667 [==============================] - 219s 329ms/step - loss: 0.3994
- dice_coef: 0.6299 - val_loss: 0.4466 - val_dice_coef: 0.5882

Epoch 00041: val_dice_coef improved from 0.57912 to 0.58820, saving model
to /content/weights.best.hdf5
Epoch 42/50
```

```
667/667 [==============================] - 221s 331ms/step - loss: 0.3987
- dice_coef: 0.6305 - val_loss: 0.5524 - val_dice_coef: 0.4881

Epoch 00042: val_dice_coef did not improve from 0.58820
Epoch 43/50
667/667 [==============================] - 221s 332ms/step - loss: 0.4020
- dice_coef: 0.6272 - val_loss: 0.4623 - val_dice_coef: 0.5736

Epoch 00043: val_dice_coef did not improve from 0.58820
Epoch 44/50
667/667 [==============================] - 223s 335ms/step - loss: 0.4019
- dice_coef: 0.6273 - val_loss: 0.4785 - val_dice_coef: 0.5597

Epoch 00044: val_dice_coef did not improve from 0.58820
Epoch 45/50
667/667 [==============================] - 221s 331ms/step - loss: 0.4001
- dice_coef: 0.6291 - val_loss: 0.4919 - val_dice_coef: 0.5482

Epoch 00045: val_dice_coef did not improve from 0.58820
Epoch 46/50
667/667 [==============================] - 220s 330ms/step - loss: 0.3934
- dice_coef: 0.6351 - val_loss: 0.4582 - val_dice_coef: 0.5753

Epoch 00046: val_dice_coef did not improve from 0.58820
Epoch 47/50
667/667 [==============================] - 220s 329ms/step - loss: 0.3943
- dice_coef: 0.6344 - val_loss: 0.4543 - val_dice_coef: 0.5809

Epoch 00047: val_dice_coef did not improve from 0.58820
Epoch 48/50
667/667 [==============================] - 220s 329ms/step - loss: 0.3905
- dice_coef: 0.6381 - val_loss: 0.4447 - val_dice_coef: 0.5897

Epoch 00048: val_dice_coef improved from 0.58820 to 0.58972, saving model
to /content/weights.best.hdf5
Epoch 49/50
667/667 [==============================] - 220s 329ms/step - loss: 0.3942
- dice_coef: 0.6345 - val_loss: 0.4480 - val_dice_coef: 0.5867

Epoch 00049: val_dice_coef did not improve from 0.58972
Epoch 50/50
667/667 [==============================] - 219s 329ms/step - loss: 0.3922
- dice_coef: 0.6364 - val_loss: 0.4546 - val_dice_coef: 0.5802

Epoch 00050: val_dice_coef did not improve from 0.58972
```

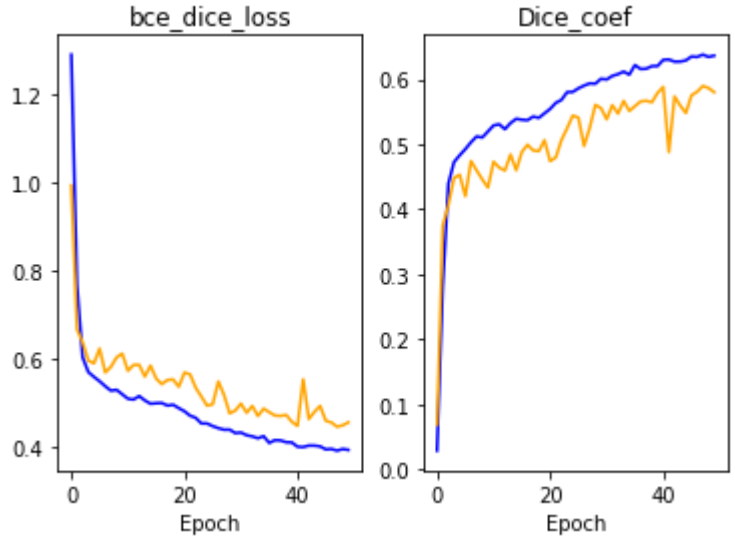
1.3 Plots on training & validation results

Loss function & metric plots

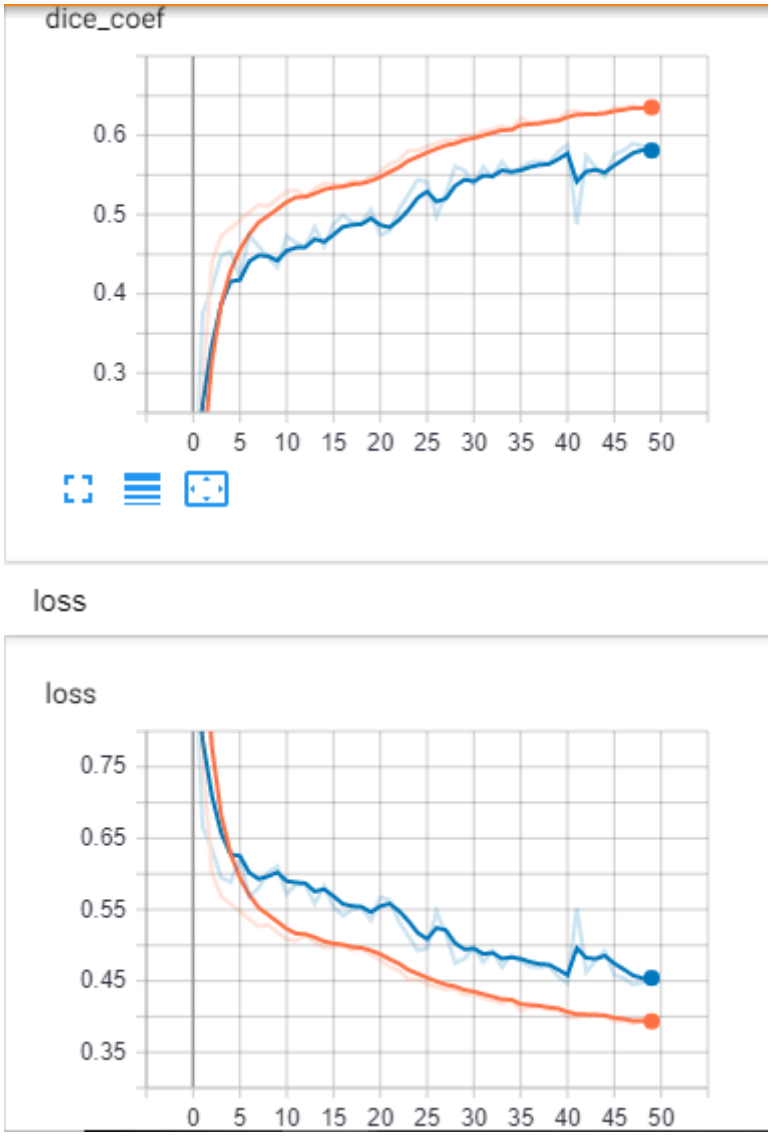
Please refer to Training Plots in utility functions section.

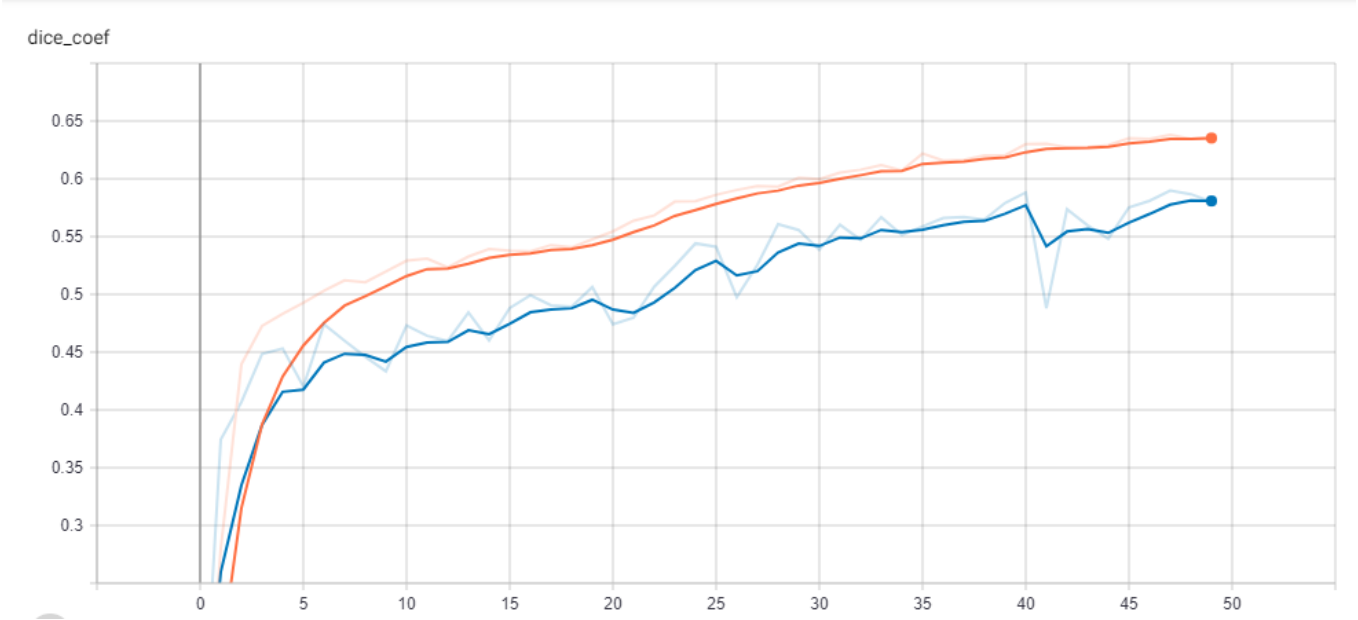
In [0]:

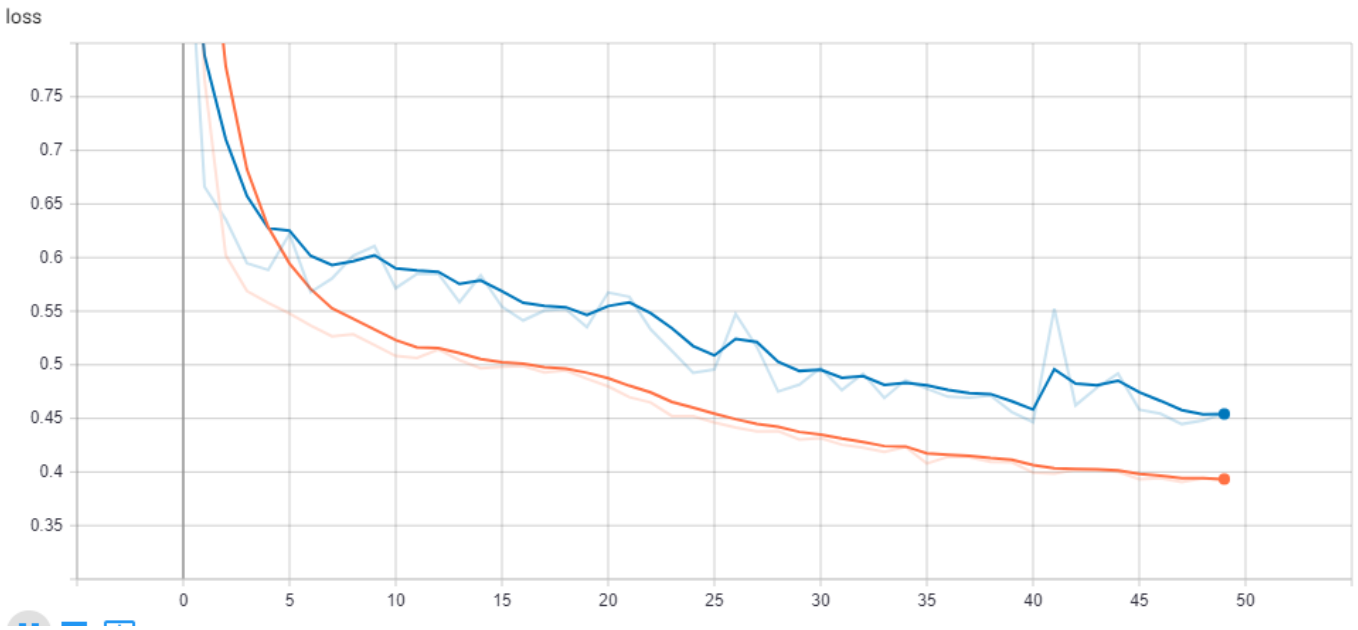
```
plot(history)
```



Tensorboard images







1.4 Model Testing

Loading the best model for evaluation

In [0]:

```
from keras.models import load_model
dependencies = {'bce_dice_loss':bce_dice_loss,'dice_coef':dice_coef,}
model_best = load_model('/content/best_model_unet.h5',custom_objects=dependencies)
```

Evaluating on validation images

In [0]:

```
evals= model_best.evaluate(valid_batches,verbose=1)
```

```
117/117 [==============================] - 36s 306ms/step
```

In [0]:

```
print('Validation set evaluation score:')
print('bce_dice loss:',evals[0])
print('dice_coeff:',evals[1])
```

```
Validation set evaluation score:
bce_dice loss: 0.4407530933873266
dice_coeff: 0.5941144448314977
```

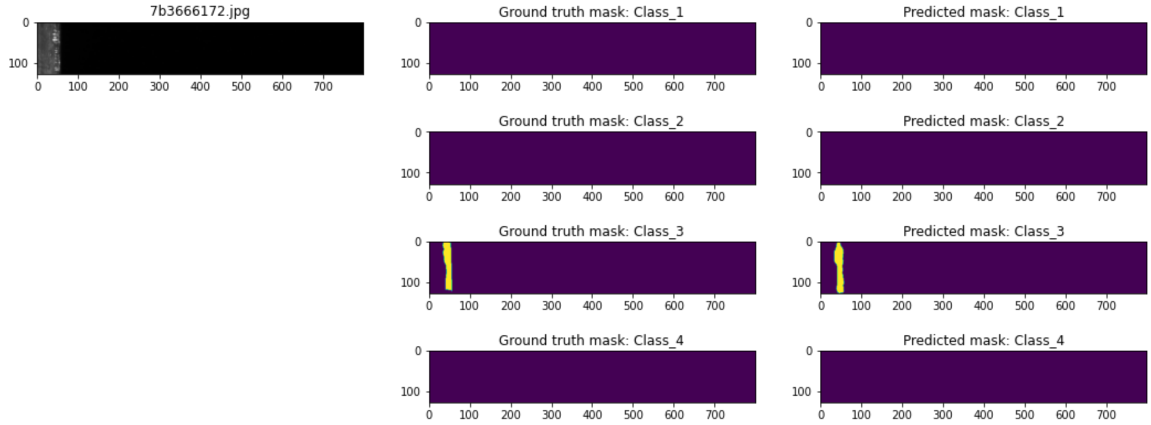
1.5 Defects visualization

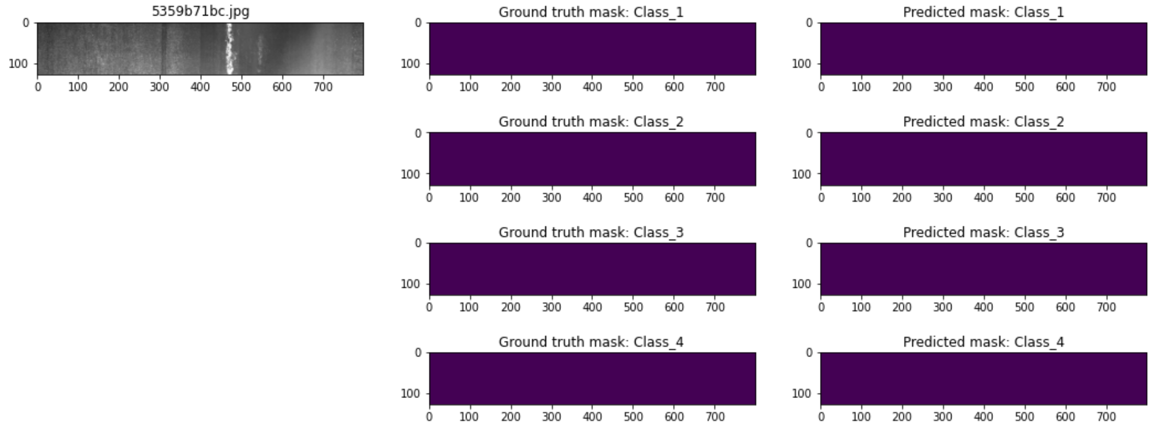
Please refer to the "visualize_defects" function in Utility functions section.

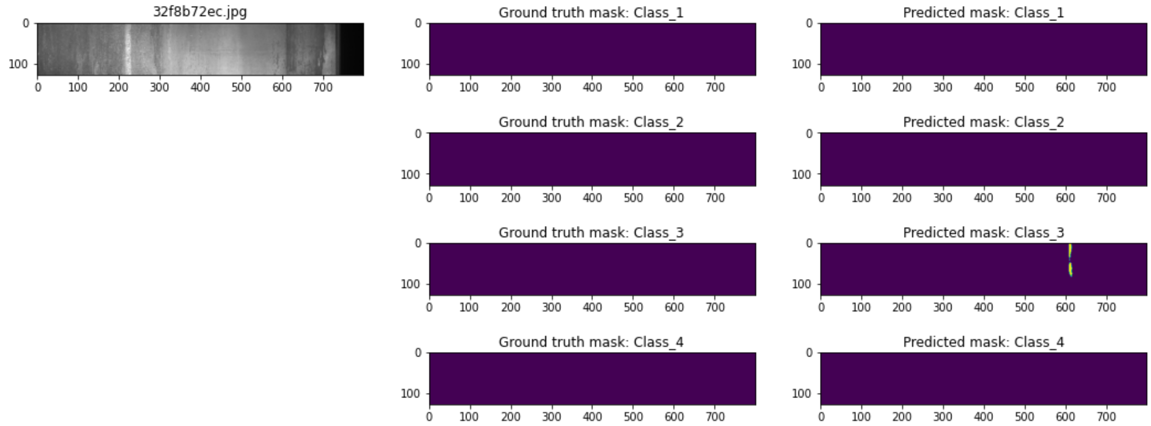
Training set

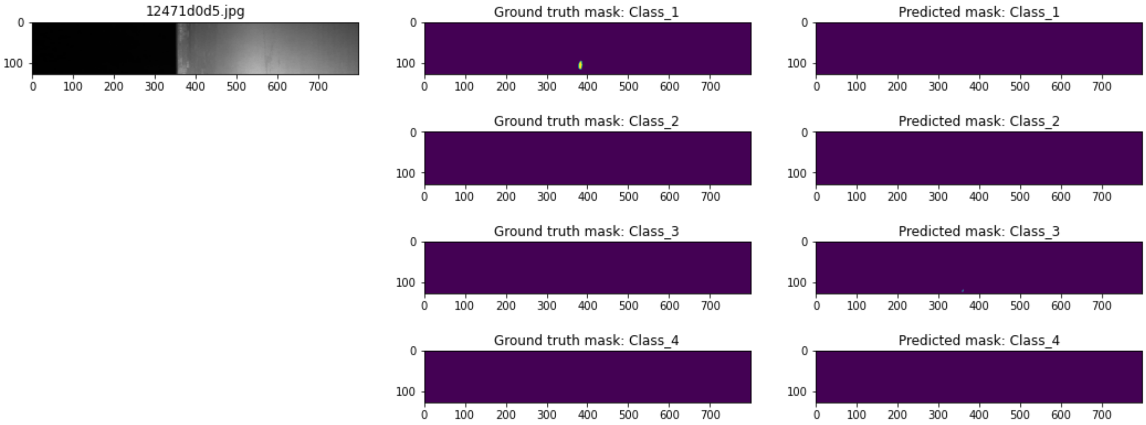
In [0]:

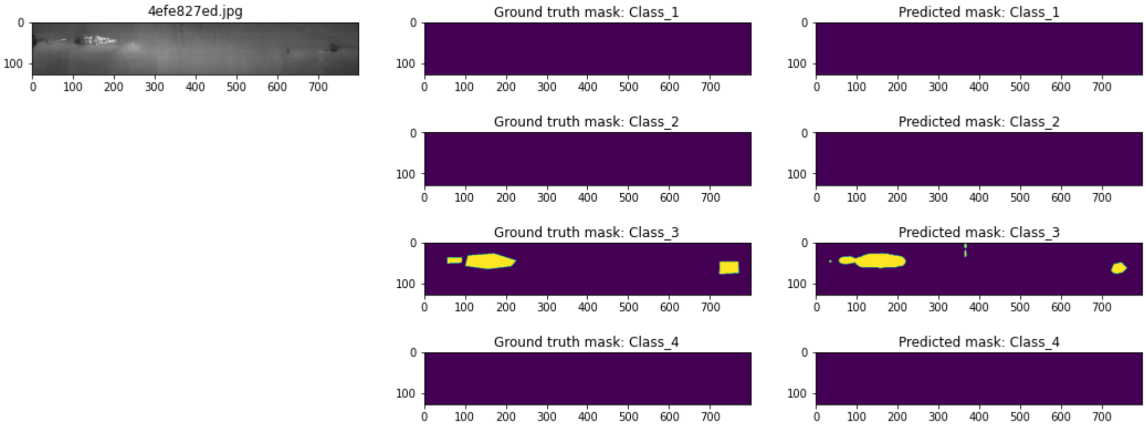
```
visualize_defects(train_data,model_best)
```


7b3666172.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


5359b71bc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

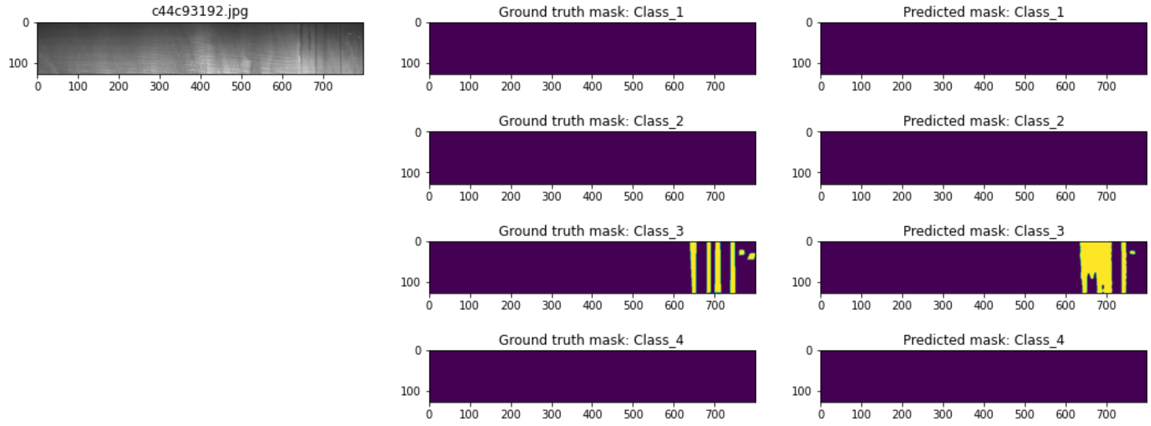

32f8b72ec.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

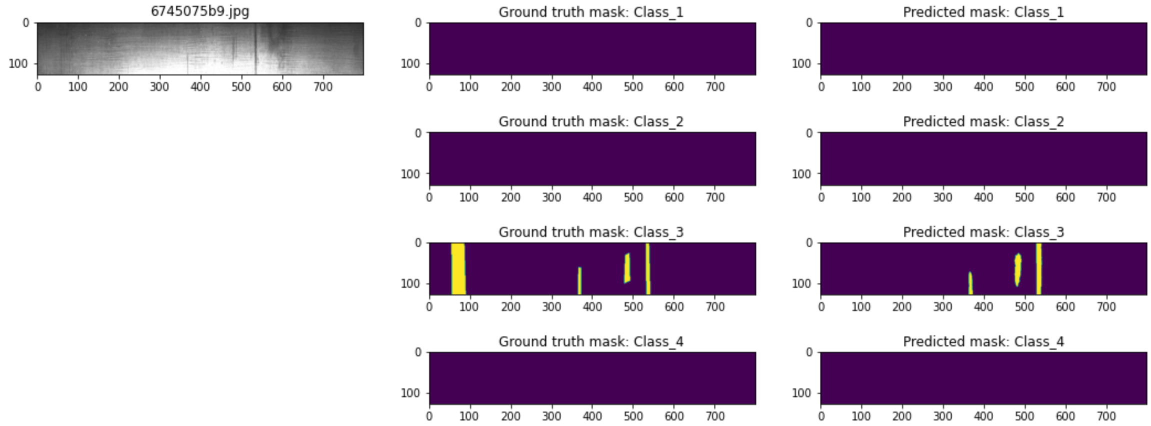

12471d0d5.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

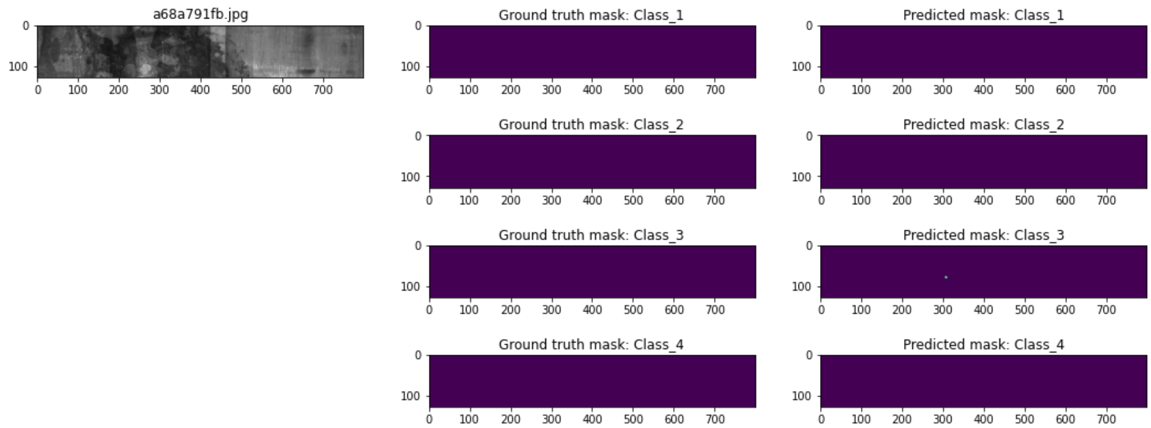

4efe827ed.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

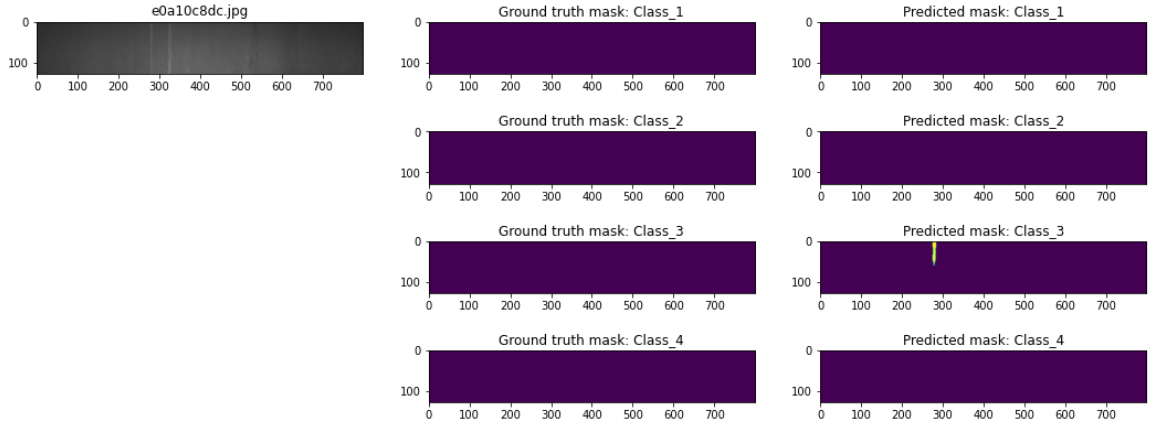
In [0]:

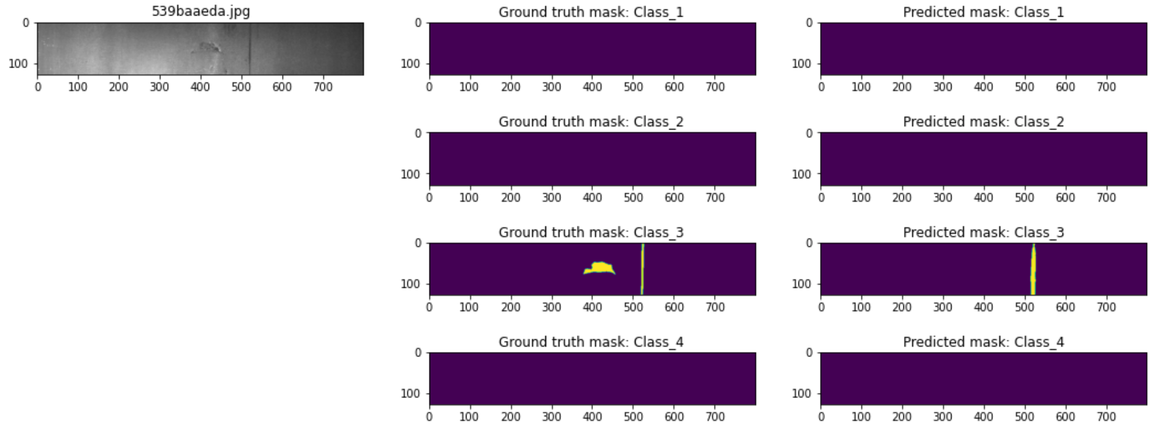
```
visualize_defects(train_data,model_best)
```


c44c93192.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


6745075b9.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

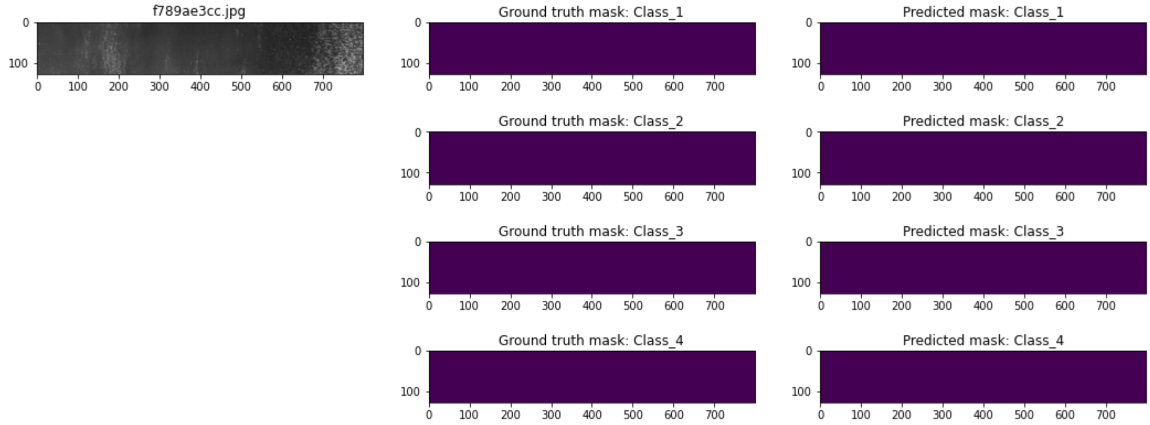

a68a791fb.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

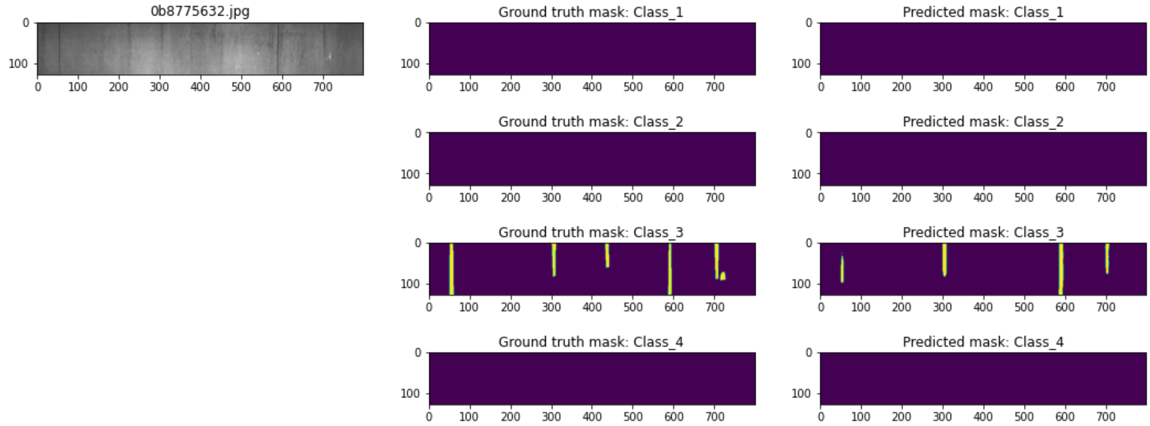

e0a10c8dc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

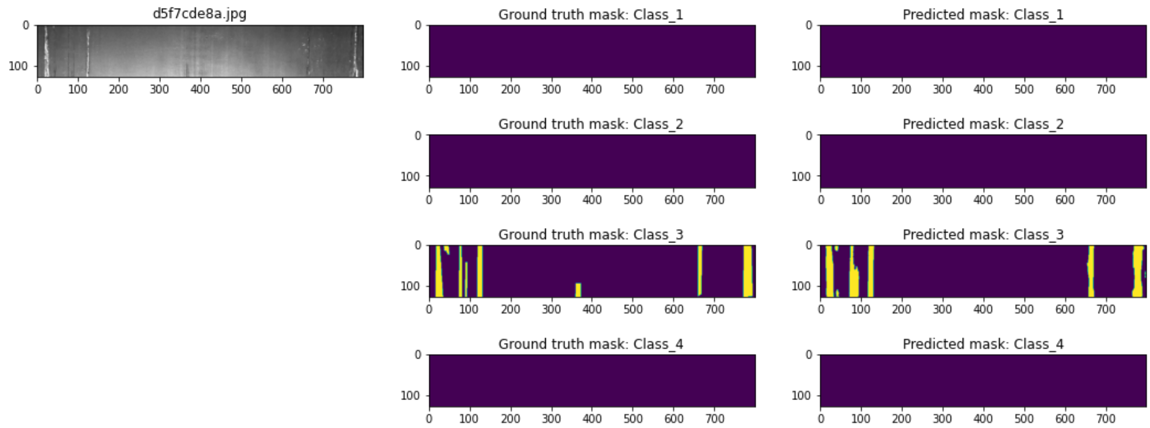

539baaeda.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

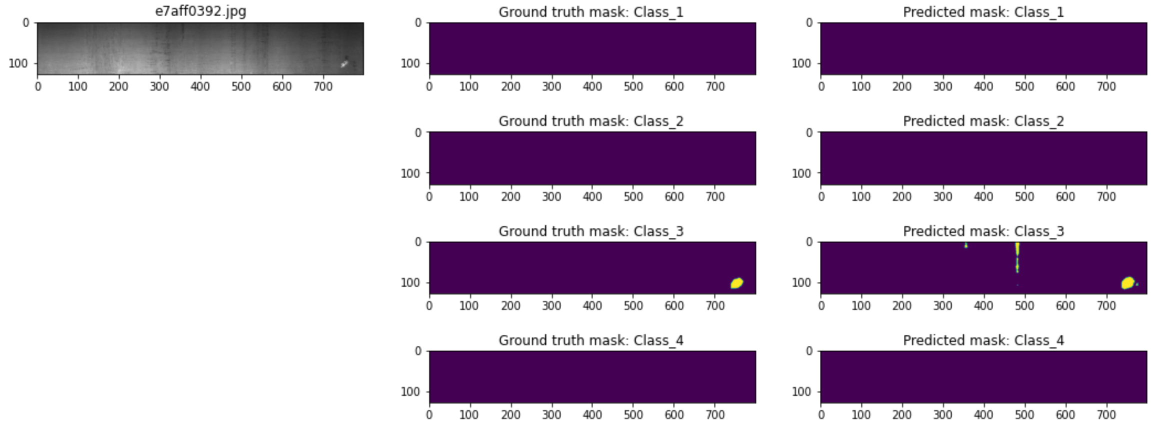
In [0]:

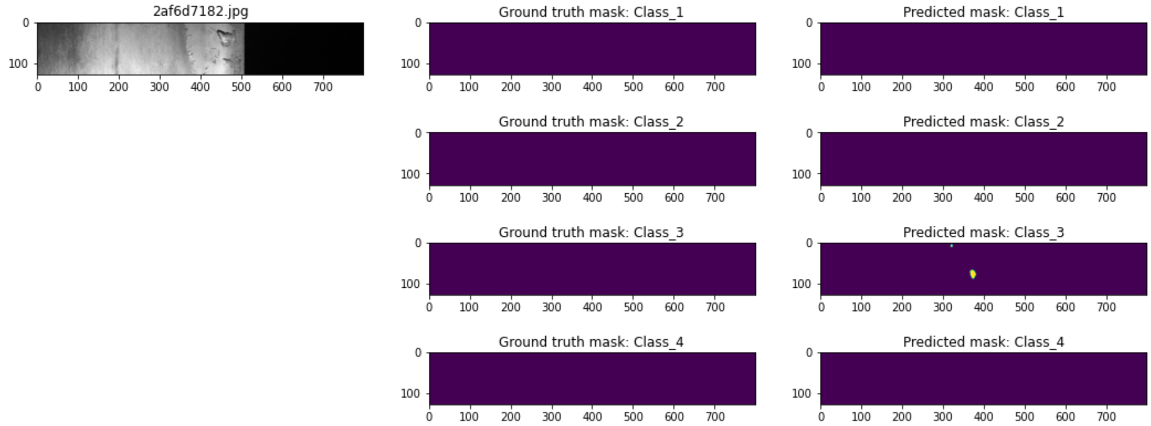
```
visualize_defects(train_data,model_best)
```


f789ae3cc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


0b8775632.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


d5f7cde8a.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

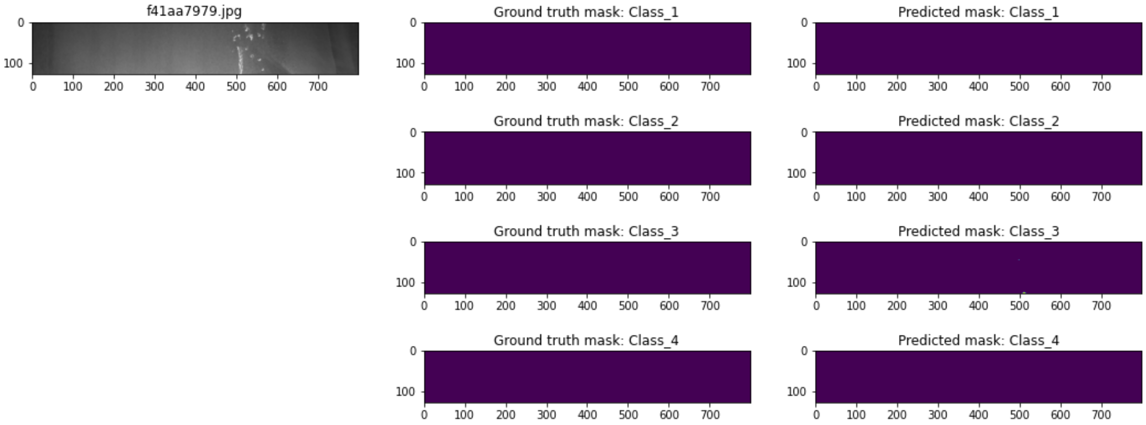


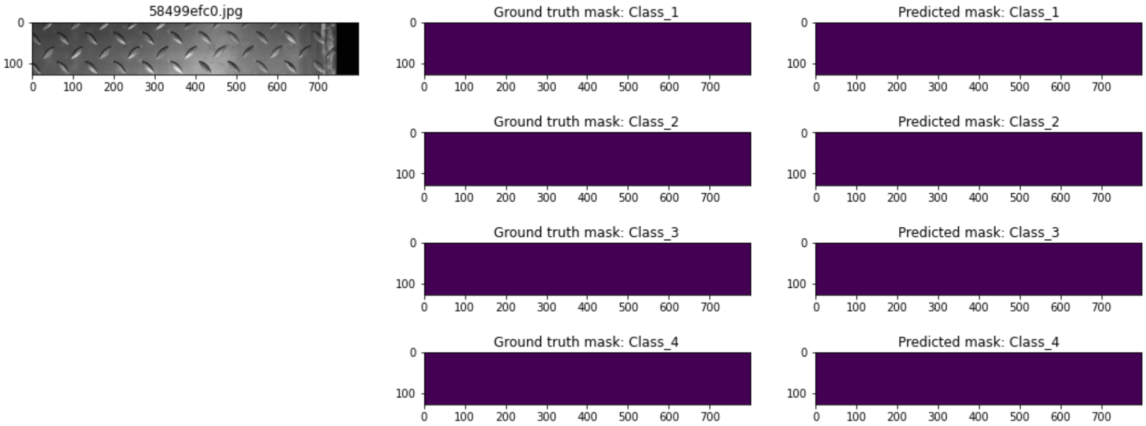


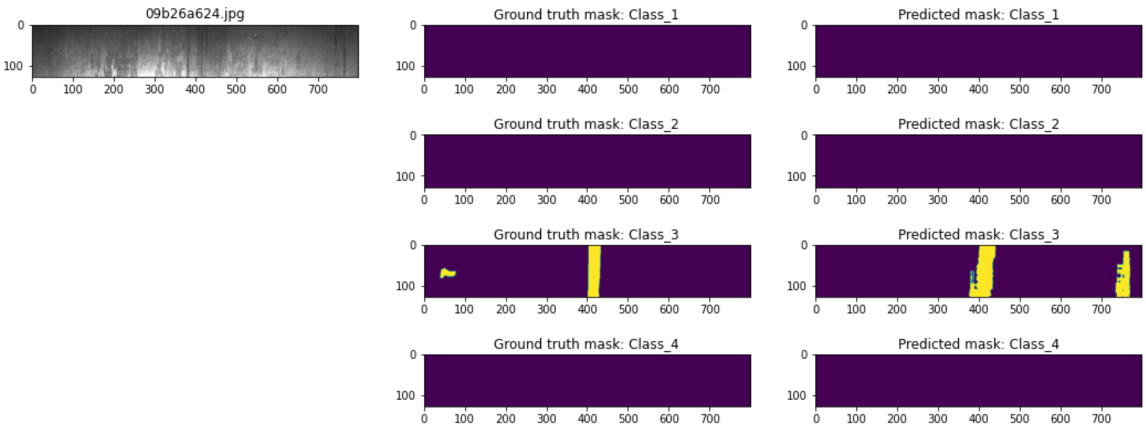
Validation set

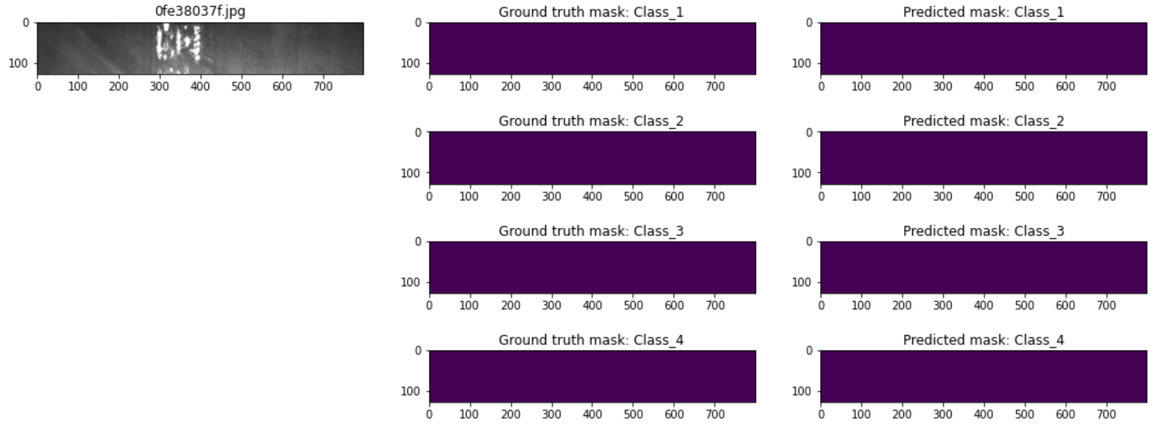
In [0]:

```
visualize_defects(cv_data,model_best)
```

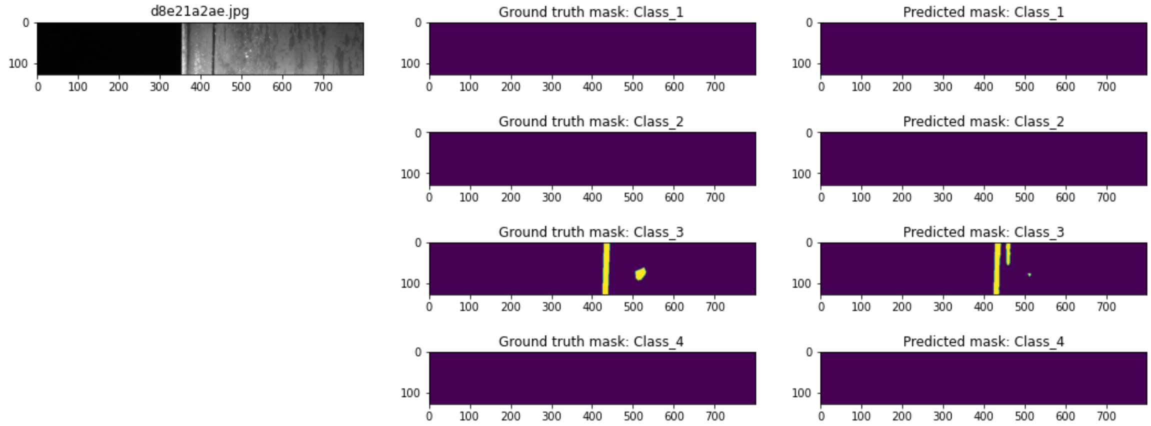

f41aa7979.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


58499efc0.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


09b26a624.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


0fe38037f.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

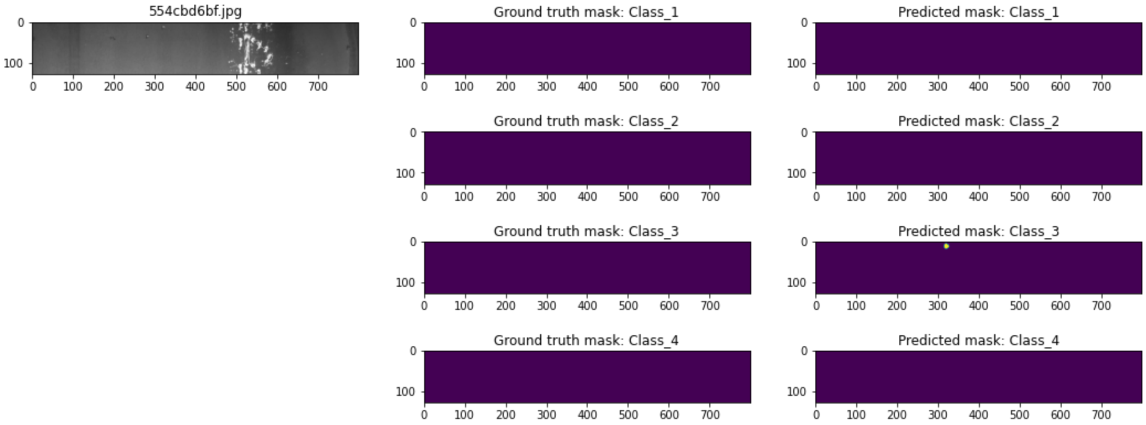
```
--------------------------------------------------------------------------------------------------------------
```

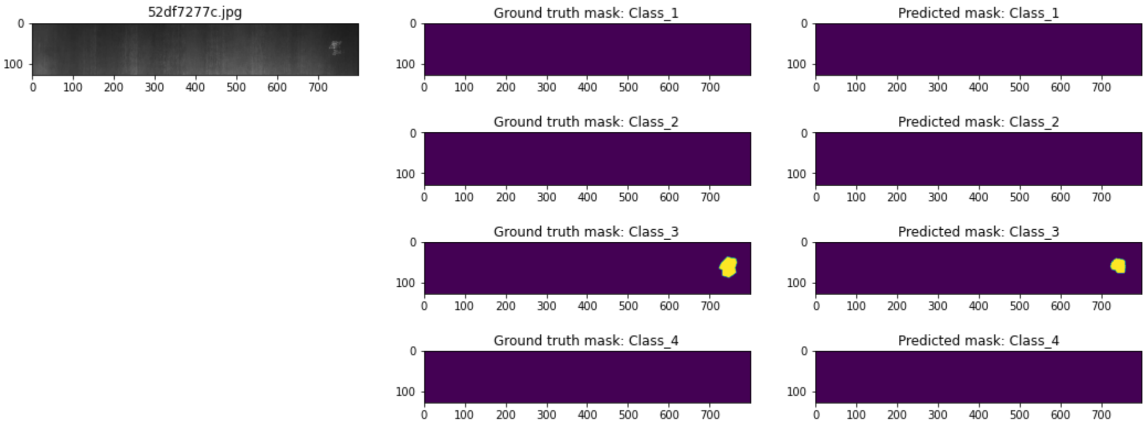

d8e21a2ae.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

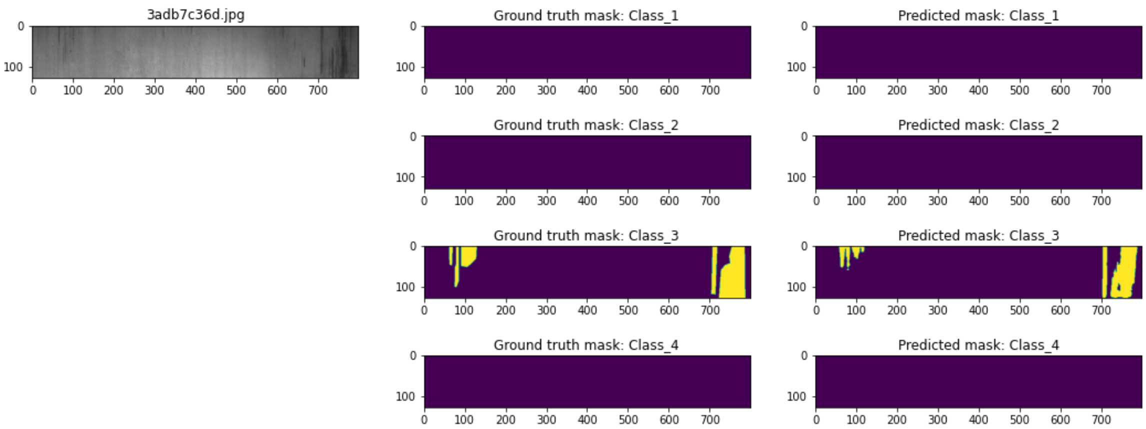
```
--------------------------------------------------------------------------------------------------------------
```

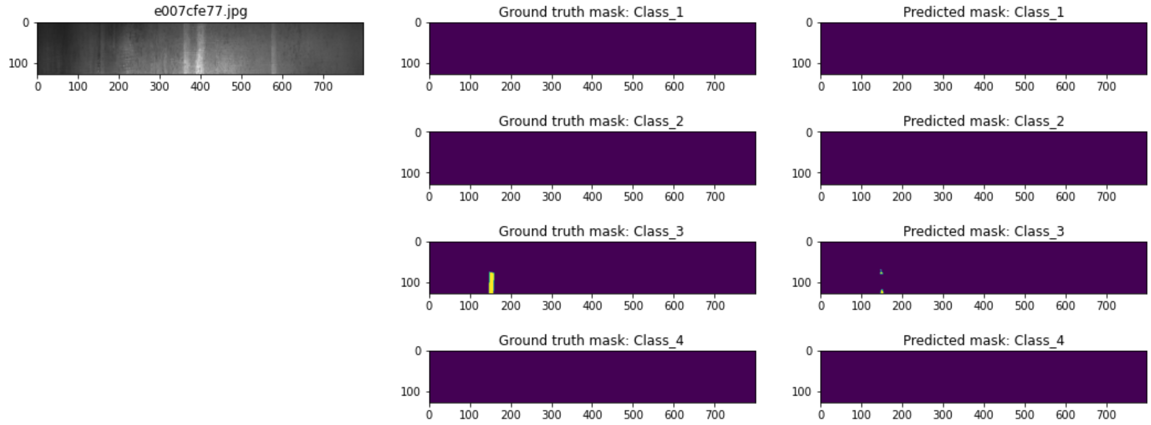
In [0]:

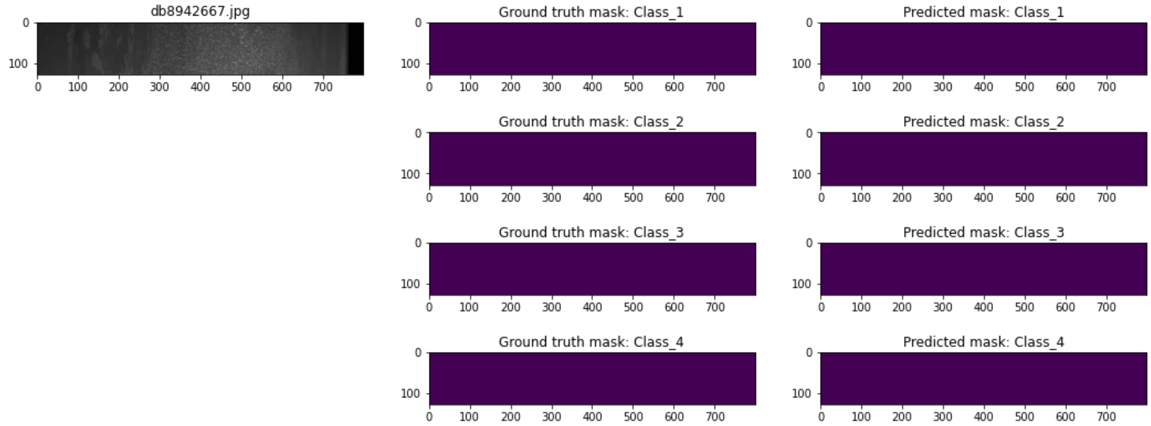
```
visualize_defects(cv_data,model_best)
```


554cbd6bf.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


52df7277c.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

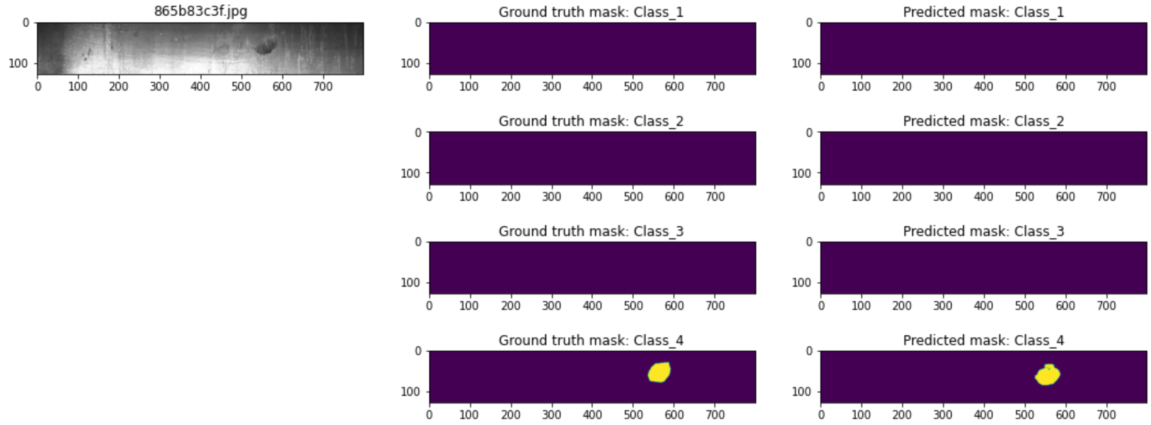

3adb7c36d.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


e007cfe77.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

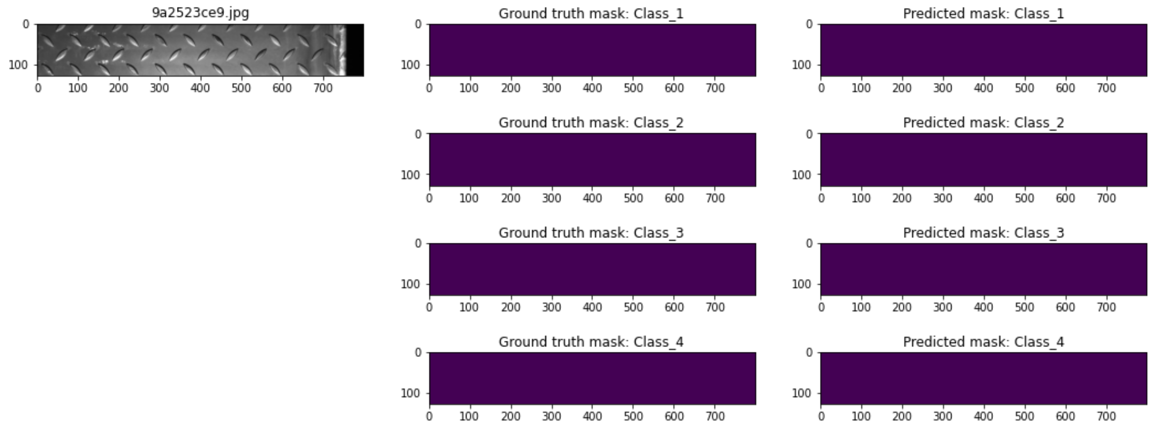

db8942667.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

In [0]:

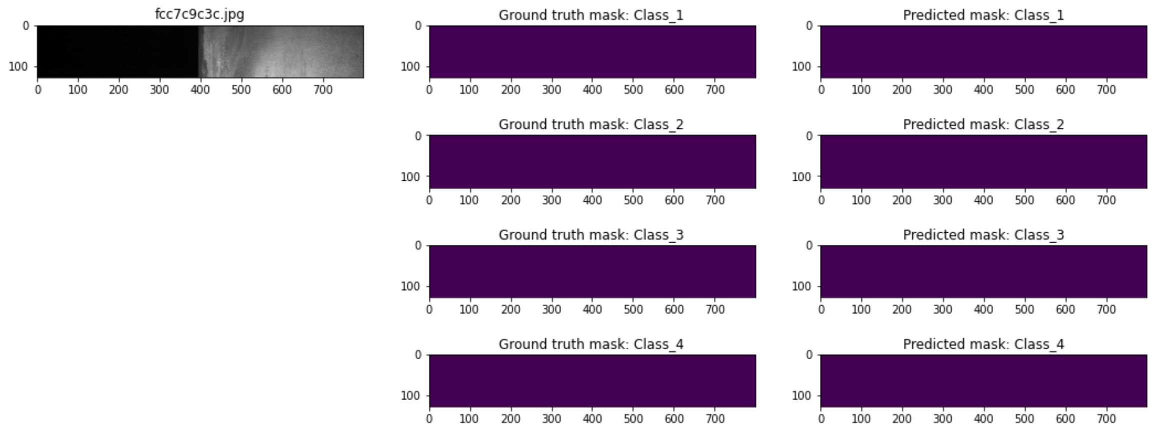
```
visualize_defects(cv_data,model_best)
```


865b83c3f.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

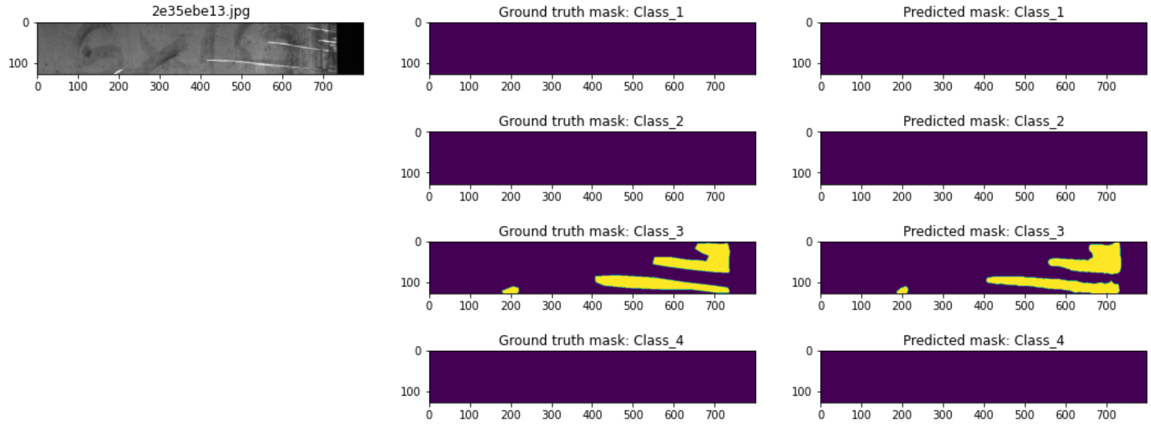
```
-----------------------------------------------------------------------------------
-----------------------------
```


9a2523ce9.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

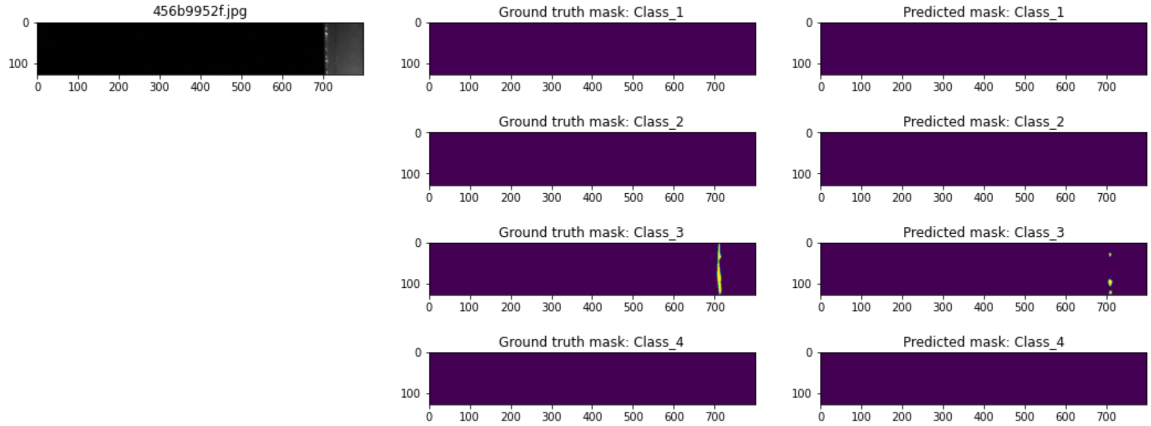
```
-----------------------------------------------------------------------------------
-----------------------------
```


fcc7c9c3c.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

```
-----------------------------------------------------------------------------------
-----------------------------
```




```
--------------------------------------------------------------------------------------------------
```




```
--------------------------------------------------------------------------------------------------
```

1.6 Predicting defects on raw test images

Predicting on raw test images by resizing images to 128x800 dimensions in order to apply the model

In [0]:

```
# Predicting on test images
from tqdm import tqdm
import cv2
data_path = '/content/' + 'test_images/'
files = list(os.listdir(data_path))
rle_lst = [] #list to store defect in run length encoding format
img_classId= [] #list to store Image ID + classId
for f in tqdm(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_best.predict(X)
    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))
```

100%|██████████| 5506/5506 [01:29<00:00, 61.81it/s]

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission_unet_128X800.csv', index=False)
```

With this submission in kaggle, I got a private dice coefficient score of 0.79814 & a public score of 0.78045.

Redefining the unet architecture for original image size of 256x1600

In [0]:

```
input_img = Input((256,1600,3), name='img')
model = get_unet(input_img, n_filters=8, dropout=0.2, batchnorm=True)
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model.summary()
```

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
img (InputLayer)             (None, 256, 1600, 3)      0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 256, 1600, 8)      224
img[0][0]
_________________________________________________________________
batch_normalization_37 (BatchNo (None, 256, 1600, 8)      32
conv2d_39[0][0]
_________________________________________________________________
activation_37 (Activation)   (None, 256, 1600, 8)      0
batch_normalization_37[0][0]
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 256, 1600, 8)      584
activation_37[0][0]
_________________________________________________________________
batch_normalization_38 (BatchNo (None, 256, 1600, 8)      32
conv2d_40[0][0]
_________________________________________________________________
activation_38 (Activation)   (None, 256, 1600, 8)      0
batch_normalization_38[0][0]
_________________________________________________________________
max_pooling2d_9 (MaxPooling2D)  (None, 128, 800, 8)   0
activation_38[0][0]
_________________________________________________________________
dropout_17 (Dropout)         (None, 128, 800, 8)   0
max_pooling2d_9[0][0]
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 128, 800, 16)      1168
dropout_17[0][0]
_________________________________________________________________
batch_normalization_39 (BatchNo (None, 128, 800, 16)      64
conv2d_41[0][0]
_________________________________________________________________
activation_39 (Activation)   (None, 128, 800, 16)      0
batch_normalization_39[0][0]
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 128, 800, 16)      2320
activation_39[0][0]
_________________________________________________________________
batch_normalization_40 (BatchNo (None, 128, 800, 16)      64
conv2d_42[0][0]
_________________________________________________________________
```

```
activation_40 (Activation)      (None, 128, 800, 16) 0           batch_nor
malization_40[0][0]
__________________________________________________________________________
_____________________
max_pooling2d_10 (MaxPooling2D) (None, 64, 400, 16)  0           activatio
n_40[0][0]
__________________________________________________________________________
_____________________
dropout_18 (Dropout)            (None, 64, 400, 16)  0           max_pooli
ng2d_10[0][0]
__________________________________________________________________________
_____________________
conv2d_43 (Conv2D)              (None, 64, 400, 32)  4640        dropout_1
8[0][0]
__________________________________________________________________________
_____________________
batch_normalization_41 (BatchNo (None, 64, 400, 32)  128         conv2d_43
[0][0]
__________________________________________________________________________
_____________________
activation_41 (Activation)      (None, 64, 400, 32)  0           batch_nor
malization_41[0][0]
__________________________________________________________________________
_____________________
conv2d_44 (Conv2D)              (None, 64, 400, 32)  9248        activatio
n_41[0][0]
__________________________________________________________________________
_____________________
batch_normalization_42 (BatchNo (None, 64, 400, 32)  128         conv2d_44
[0][0]
__________________________________________________________________________
_____________________
activation_42 (Activation)      (None, 64, 400, 32)  0           batch_nor
malization_42[0][0]
__________________________________________________________________________
_____________________
max_pooling2d_11 (MaxPooling2D) (None, 32, 200, 32)  0           activatio
n_42[0][0]
__________________________________________________________________________
_____________________
dropout_19 (Dropout)            (None, 32, 200, 32)  0           max_pooli
ng2d_11[0][0]
__________________________________________________________________________
_____________________
conv2d_45 (Conv2D)              (None, 32, 200, 64)  18496       dropout_1
9[0][0]
__________________________________________________________________________
_____________________
batch_normalization_43 (BatchNo (None, 32, 200, 64)  256         conv2d_45
[0][0]
__________________________________________________________________________
_____________________
activation_43 (Activation)      (None, 32, 200, 64)  0           batch_nor
malization_43[0][0]
__________________________________________________________________________
_____________________
conv2d_46 (Conv2D)              (None, 32, 200, 64)  36928       activatio
n_43[0][0]
__________________________________________________________________________
_____________________
batch_normalization_44 (BatchNo (None, 32, 200, 64)  256         conv2d_46
```

```
[0][0]
__________________________________________________________________________________________
_____________________
activation_44 (Activation)      (None, 32, 200, 64)  0           batch_nor
malization_44[0][0]
__________________________________________________________________________________________
_____________________
max_pooling2d_12 (MaxPooling2D) (None, 16, 100, 64)  0           activatio
n_44[0][0]
__________________________________________________________________________________________
_____________________
dropout_20 (Dropout)            (None, 16, 100, 64)  0           max_pooli
ng2d_12[0][0]
__________________________________________________________________________________________
_____________________
conv2d_47 (Conv2D)              (None, 16, 100, 128) 73856       dropout_2
0[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_45 (BatchNo (None, 16, 100, 128) 512         conv2d_47
[0][0]
__________________________________________________________________________________________
_____________________
activation_45 (Activation)      (None, 16, 100, 128) 0           batch_nor
malization_45[0][0]
__________________________________________________________________________________________
_____________________
conv2d_48 (Conv2D)              (None, 16, 100, 128) 147584      activatio
n_45[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_46 (BatchNo (None, 16, 100, 128) 512         conv2d_48
[0][0]
__________________________________________________________________________________________
_____________________
activation_46 (Activation)      (None, 16, 100, 128) 0           batch_nor
malization_46[0][0]
__________________________________________________________________________________________
_____________________
conv2d_transpose_9 (Conv2DTrans (None, 32, 200, 64)  73792       activatio
n_46[0][0]
__________________________________________________________________________________________
_____________________
concatenate_9 (Concatenate)     (None, 32, 200, 128) 0           conv2d_tr
anspose_9[0][0]
                                                                 activatio
n_44[0][0]
__________________________________________________________________________________________
_____________________
dropout_21 (Dropout)            (None, 32, 200, 128) 0           concatena
te_9[0][0]
__________________________________________________________________________________________
_____________________
conv2d_49 (Conv2D)              (None, 32, 200, 64)  73792       dropout_2
1[0][0]
__________________________________________________________________________________________
_____________________
batch_normalization_47 (BatchNo (None, 32, 200, 64)  256         conv2d_49
[0][0]
__________________________________________________________________________________________
_____________________
```

```
activation_47 (Activation)      (None, 32, 200, 64)  0           batch_nor
malization_47[0][0]
__________________________________________________________________________
_____________________
conv2d_50 (Conv2D)              (None, 32, 200, 64)  36928       activatio
n_47[0][0]
__________________________________________________________________________
_____________________
batch_normalization_48 (BatchNo (None, 32, 200, 64)  256         conv2d_50
[0][0]
__________________________________________________________________________
_____________________
activation_48 (Activation)      (None, 32, 200, 64)  0           batch_nor
malization_48[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_10 (Conv2DTran (None, 64, 400, 32)  18464       activatio
n_48[0][0]
__________________________________________________________________________
_____________________
concatenate_10 (Concatenate)    (None, 64, 400, 64)  0           conv2d_tr
anspose_10[0][0]
                                                                 activatio
n_42[0][0]
__________________________________________________________________________
_____________________
dropout_22 (Dropout)            (None, 64, 400, 64)  0           concatena
te_10[0][0]
__________________________________________________________________________
_____________________
conv2d_51 (Conv2D)              (None, 64, 400, 32)  18464       dropout_2
2[0][0]
__________________________________________________________________________
_____________________
batch_normalization_49 (BatchNo (None, 64, 400, 32)  128         conv2d_51
[0][0]
__________________________________________________________________________
_____________________
activation_49 (Activation)      (None, 64, 400, 32)  0           batch_nor
malization_49[0][0]
__________________________________________________________________________
_____________________
conv2d_52 (Conv2D)              (None, 64, 400, 32)  9248        activatio
n_49[0][0]
__________________________________________________________________________
_____________________
batch_normalization_50 (BatchNo (None, 64, 400, 32)  128         conv2d_52
[0][0]
__________________________________________________________________________
_____________________
activation_50 (Activation)      (None, 64, 400, 32)  0           batch_nor
malization_50[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_11 (Conv2DTran (None, 128, 800, 16) 4624        activatio
n_50[0][0]
__________________________________________________________________________
_____________________
concatenate_11 (Concatenate)    (None, 128, 800, 32) 0           conv2d_tr
anspose_11[0][0]
                                                                 activatio
```

```
n_40[0][0]
__________________________________________________________________________________________________
__________________________
dropout_23 (Dropout)            (None, 128, 800, 32) 0           concatena
te_11[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_53 (Conv2D)              (None, 128, 800, 16) 4624        dropout_2
3[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_51 (BatchNo (None, 128, 800, 16) 64          conv2d_53
[0][0]
__________________________________________________________________________________________________
__________________________
activation_51 (Activation)      (None, 128, 800, 16) 0           batch_nor
malization_51[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_54 (Conv2D)              (None, 128, 800, 16) 2320        activatio
n_51[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_52 (BatchNo (None, 128, 800, 16) 64          conv2d_54
[0][0]
__________________________________________________________________________________________________
__________________________
activation_52 (Activation)      (None, 128, 800, 16) 0           batch_nor
malization_52[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_transpose_12 (Conv2DTran (None, 256, 1600, 8) 1160        activatio
n_52[0][0]
__________________________________________________________________________________________________
__________________________
concatenate_12 (Concatenate)    (None, 256, 1600, 16 0           conv2d_tr
anspose_12[0][0]
                                                                 activatio
n_38[0][0]
__________________________________________________________________________________________________
__________________________
dropout_24 (Dropout)            (None, 256, 1600, 16 0           concatena
te_12[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_55 (Conv2D)              (None, 256, 1600, 8) 1160        dropout_2
4[0][0]
__________________________________________________________________________________________________
__________________________
batch_normalization_53 (BatchNo (None, 256, 1600, 8) 32          conv2d_55
[0][0]
__________________________________________________________________________________________________
__________________________
activation_53 (Activation)      (None, 256, 1600, 8) 0           batch_nor
malization_53[0][0]
__________________________________________________________________________________________________
__________________________
conv2d_56 (Conv2D)              (None, 256, 1600, 8) 584         activatio
n_53[0][0]
__________________________________________________________________________________________________
__________________________
```

```
batch_normalization_54 (BatchNo (None, 256, 1600, 8) 32          conv2d_56
[0][0]
__________________________________________________________________________
_____________________
activation_54 (Activation)      (None, 256, 1600, 8) 0           batch_nor
malization_54[0][0]
__________________________________________________________________________
_____________________
conv2d_57 (Conv2D)              (None, 256, 1600, 4) 36          activatio
n_54[0][0]
==========================================================================
=======================
Total params: 543,188
Trainable params: 541,716
Non-trainable params: 1,472
__________________________________________________________________________
_____________________
```

In [0]:

```
model.set_weights(model_best.get_weights()) #transfering the weights of 128x800 model
```

Predicting on full 256x1600 raw test images

In [0]:

```
# Predicting on test images
from tqdm import tqdm
import cv2
data_path = '/content/' + 'test_images/'
files = list(os.listdir(data_path))
rle_lst = [] #list to store defect in run length encoding format
img_classId= [] #list to store Image ID + classId

for f in tqdm(files):
    X = np.empty((1,256,1600,3),dtype=np.float32)
    img = cv2.imread(data_path + f)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    X[0,] = img
    mask = model.predict(X)
    rle_m = np.empty((256,1600),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))
```

100%|██████████| 5506/5506 [04:12<00:00, 21.80it/s]

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission_unet_256_1600.csv', index=False)
```

With this submission, I got a private dice coefficient score of 0.81171 & a public score of 0.80041.

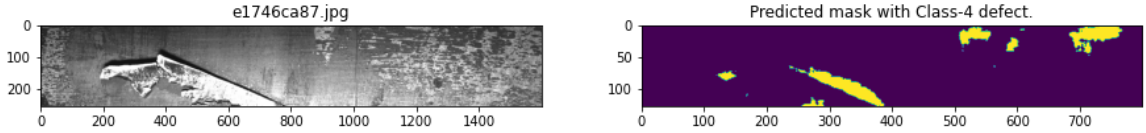
- This task of predicting on full images was inspired from this kernel: https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114321 (https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114321)
- The author of the kernel had proof of better scores when defects were predicted on 256x1600 instead of 128x800
- This strategy worked out for me as I got a higher score when predicted on full images compared to half images
- **Henceforth, for other architectures, I will be predicting on 256x1800 images instead of 128x800**

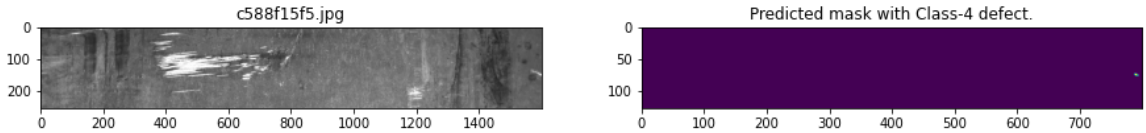
1.7 Visualizing defects of raw test images(256x1600)

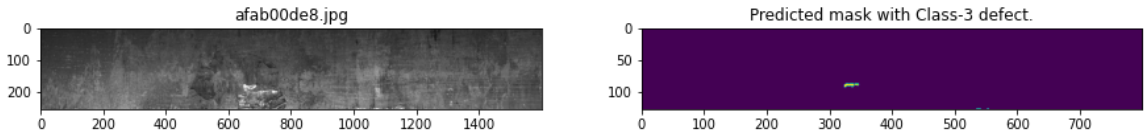
Please refer to the "visualize_defects_test" function in Utility functions section.

In [0]:

```
visualize_defects_test(output_df,5)
```





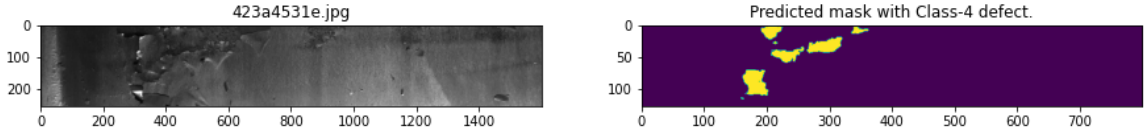


In [0]:

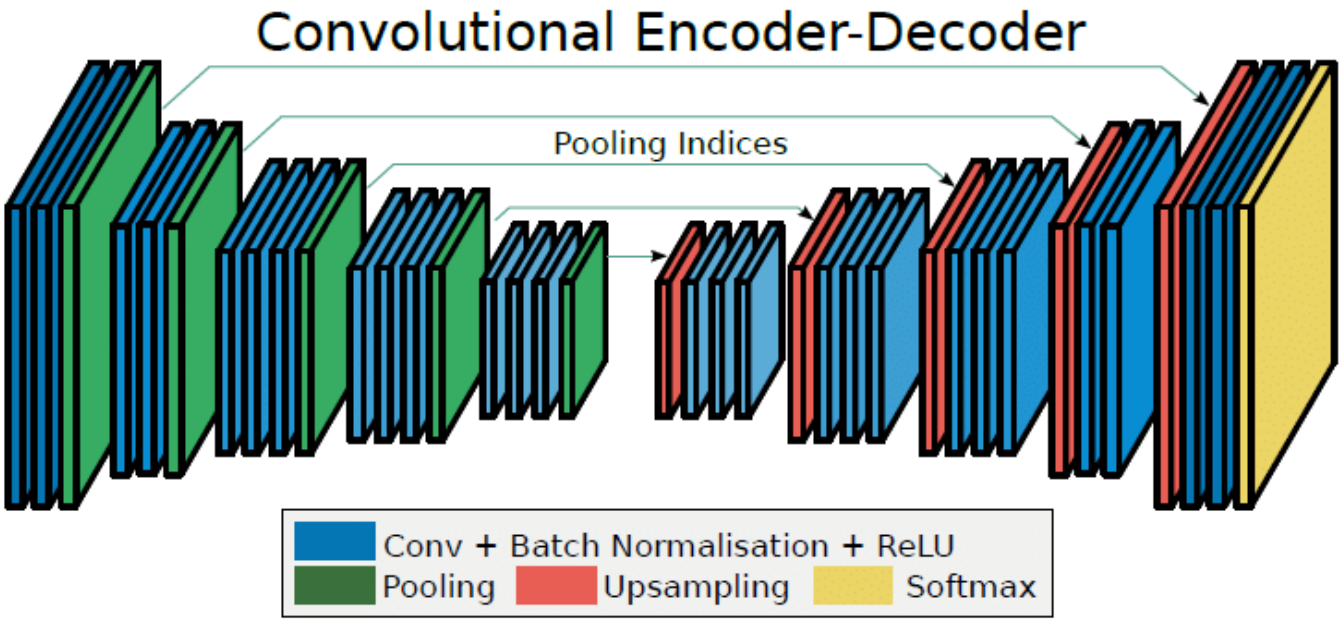
```
visualize_defects_test(output_df,5)
```

In [0]:

```
visualize_defects_test(output_df,5)
```



2.0 Segmentation Net(Segnet)



References-

- https://github.com/imlab-uiip/keras-segnet (https://github.com/imlab-uiip/keras-segnet)
- https://arxiv.org/pdf/1505.04366.pdf (https://arxiv.org/pdf/1505.04366.pdf)

In [0]:

```
def two_conv2d_blocks(input_tensor, n_filters, kernel_size = 3, batchnorm = True):
    """Function containing 2 convolutional layers with the parameters passed to it"""
    # first layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
               kernel_initializer = 'he_normal', padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # second layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
               kernel_initializer = 'he_normal', padding = 'same')(x)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    return x
```

In [0]:

```
def single_conv2d_block(input_tensor, n_filters, kernel_size = 3, batchnorm = True):
    """Function containing 1 convolutional layer with the parameters passed to it"""
    # first layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size, kernel_size),\
              kernel_initializer = 'he_normal', padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    return x
```

In [0]:

```
def get_segnet(input_img, n_filters, dropout, batchnorm):

    """Function to define the Segnet architecture"""

    # Encoder Path
    c1 = two_conv2d_blocks(input_img, n_filters * 1, kernel_size = 3, batchnorm = batch
norm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)

    c2 = two_conv2d_blocks(p1, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)

    c3 = two_conv2d_blocks(p2, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)
    c3_1 = single_conv2d_block(c3, n_filters * 4, kernel_size = 3, batchnorm = batchnor
m)
    p3 = MaxPooling2D((2, 2))(c3_1)
    p3 = Dropout(dropout)(p3)

    c4 = two_conv2d_blocks(p3, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    c4_1= single_conv2d_block(c4, n_filters * 8, kernel_size = 3, batchnorm = batchnorm
)
    p4 = MaxPooling2D((2, 2))(c4_1)
    p4 = Dropout(dropout)(p4)

    c5 = two_conv2d_blocks(p4, n_filters = n_filters * 8, kernel_size = 3, batchnorm =
batchnorm)
    c5_1= single_conv2d_block(c5, n_filters * 8, kernel_size = 3, batchnorm = batchnorm
)
    p5 = MaxPooling2D((2, 2))(c5_1)
    p5 = Dropout(dropout)(p5)

    # Decoder Path
    u6 = UpSampling2D()(p5)
    c6 = two_conv2d_blocks(u6,n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    c6 = single_conv2d_block(c6, n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    c6 = Dropout(dropout)(c6)

    u7 = UpSampling2D()(c6)
    c7 = two_conv2d_blocks(u7,n_filters * 8, kernel_size = 3, batchnorm = batchnorm)
    c7 = single_conv2d_block(c7, n_filters * 4, kernel_size = 3, batchnorm = batchnorm)
    c7 = Dropout(dropout)(c7)

    u8 = UpSampling2D()(c7)
    c8 = two_conv2d_blocks(u8,n_filters * 4, kernel_size = 3, batchnorm = batchnorm)
    c8 = single_conv2d_block(c8, n_filters * 2, kernel_size = 3, batchnorm = batchnorm)
    c8 = Dropout(dropout)(c8)

    u9 = UpSampling2D()(c8)
    c9 = single_conv2d_block(u9,n_filters * 2, kernel_size = 3, batchnorm = batchnorm)
    c9 = single_conv2d_block(c9, n_filters * 1, kernel_size = 3, batchnorm = batchnorm)
    c9 = Dropout(dropout)(c9)

    u10 = UpSampling2D()(c9)
    c10 = single_conv2d_block(u10,n_filters * 1, kernel_size = 3, batchnorm = batchnorm
)
    c10 = Dropout(dropout)(c10)
```

```
    outputs = Conv2D(4, (1, 1), activation='sigmoid')(c10)
    model = Model(inputs=[input_img], outputs=[outputs])
    return model
```

In [0]:

```
input_img = Input((128,800,3),name='img')
model = get_segnet(input_img, n_filters=16, dropout=0.5, batchnorm=True)
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
img (InputLayer)             (None, 128, 800, 3)       0
_________________________________________________________________
conv2d_79 (Conv2D)           (None, 128, 800, 16)      448
_________________________________________________________________
batch_normalization_76 (Batc (None, 128, 800, 16)      64
_________________________________________________________________
activation_76 (Activation)   (None, 128, 800, 16)      0
_________________________________________________________________
conv2d_80 (Conv2D)           (None, 128, 800, 16)      2320
_________________________________________________________________
batch_normalization_77 (Batc (None, 128, 800, 16)      64
_________________________________________________________________
activation_77 (Activation)   (None, 128, 800, 16)      0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 64, 400, 16)       0
_________________________________________________________________
dropout_31 (Dropout)         (None, 64, 400, 16)       0
_________________________________________________________________
conv2d_81 (Conv2D)           (None, 64, 400, 32)       4640
_________________________________________________________________
batch_normalization_78 (Batc (None, 64, 400, 32)       128
_________________________________________________________________
activation_78 (Activation)   (None, 64, 400, 32)       0
_________________________________________________________________
conv2d_82 (Conv2D)           (None, 64, 400, 32)       9248
_________________________________________________________________
batch_normalization_79 (Batc (None, 64, 400, 32)       128
_________________________________________________________________
activation_79 (Activation)   (None, 64, 400, 32)       0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 32, 200, 32)       0
_________________________________________________________________
dropout_32 (Dropout)         (None, 32, 200, 32)       0
_________________________________________________________________
conv2d_83 (Conv2D)           (None, 32, 200, 64)       18496
_________________________________________________________________
batch_normalization_80 (Batc (None, 32, 200, 64)       256
_________________________________________________________________
activation_80 (Activation)   (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_84 (Conv2D)           (None, 32, 200, 64)       36928
_________________________________________________________________
batch_normalization_81 (Batc (None, 32, 200, 64)       256
_________________________________________________________________
activation_81 (Activation)   (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_85 (Conv2D)           (None, 32, 200, 64)       36928
_________________________________________________________________
batch_normalization_82 (Batc (None, 32, 200, 64)       256
_________________________________________________________________
activation_82 (Activation)   (None, 32, 200, 64)       0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 16, 100, 64)       0
_________________________________________________________________
dropout_33 (Dropout)         (None, 16, 100, 64)       0
_________________________________________________________________
conv2d_86 (Conv2D)           (None, 16, 100, 128)      73856
```

```
batch_normalization_83 (Batc (None, 16, 100, 128)      512
activation_83 (Activation)   (None, 16, 100, 128)      0
conv2d_87 (Conv2D)           (None, 16, 100, 128)      147584
batch_normalization_84 (Batc (None, 16, 100, 128)      512
activation_84 (Activation)   (None, 16, 100, 128)      0
conv2d_88 (Conv2D)           (None, 16, 100, 128)      147584
batch_normalization_85 (Batc (None, 16, 100, 128)      512
activation_85 (Activation)   (None, 16, 100, 128)      0
max_pooling2d_19 (MaxPooling (None, 8, 50, 128)        0
dropout_34 (Dropout)         (None, 8, 50, 128)        0
conv2d_89 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_86 (Batc (None, 8, 50, 128)        512
activation_86 (Activation)   (None, 8, 50, 128)        0
conv2d_90 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_87 (Batc (None, 8, 50, 128)        512
activation_87 (Activation)   (None, 8, 50, 128)        0
conv2d_91 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_88 (Batc (None, 8, 50, 128)        512
activation_88 (Activation)   (None, 8, 50, 128)        0
max_pooling2d_20 (MaxPooling (None, 4, 25, 128)        0
dropout_35 (Dropout)         (None, 4, 25, 128)        0
up_sampling2d_16 (UpSampling (None, 8, 50, 128)        0
conv2d_92 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_89 (Batc (None, 8, 50, 128)        512
activation_89 (Activation)   (None, 8, 50, 128)        0
conv2d_93 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_90 (Batc (None, 8, 50, 128)        512
activation_90 (Activation)   (None, 8, 50, 128)        0
conv2d_94 (Conv2D)           (None, 8, 50, 128)        147584
batch_normalization_91 (Batc (None, 8, 50, 128)        512
```

```
activation_91 (Activation)   (None, 8, 50, 128)        0
_________________________________________________________________
dropout_36 (Dropout)         (None, 8, 50, 128)        0
_________________________________________________________________
up_sampling2d_17 (UpSampling (None, 16, 100, 128)      0
_________________________________________________________________
conv2d_95 (Conv2D)           (None, 16, 100, 128)      147584
_________________________________________________________________
batch_normalization_92 (Batc (None, 16, 100, 128)      512
_________________________________________________________________
activation_92 (Activation)   (None, 16, 100, 128)      0
_________________________________________________________________
conv2d_96 (Conv2D)           (None, 16, 100, 128)      147584
_________________________________________________________________
batch_normalization_93 (Batc (None, 16, 100, 128)      512
_________________________________________________________________
activation_93 (Activation)   (None, 16, 100, 128)      0
_________________________________________________________________
conv2d_97 (Conv2D)           (None, 16, 100, 64)       73792
_________________________________________________________________
batch_normalization_94 (Batc (None, 16, 100, 64)       256
_________________________________________________________________
activation_94 (Activation)   (None, 16, 100, 64)       0
_________________________________________________________________
dropout_37 (Dropout)         (None, 16, 100, 64)       0
_________________________________________________________________
up_sampling2d_18 (UpSampling (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_98 (Conv2D)           (None, 32, 200, 64)       36928
_________________________________________________________________
batch_normalization_95 (Batc (None, 32, 200, 64)       256
_________________________________________________________________
activation_95 (Activation)   (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_99 (Conv2D)           (None, 32, 200, 64)       36928
_________________________________________________________________
batch_normalization_96 (Batc (None, 32, 200, 64)       256
_________________________________________________________________
activation_96 (Activation)   (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_100 (Conv2D)          (None, 32, 200, 32)       18464
_________________________________________________________________
batch_normalization_97 (Batc (None, 32, 200, 32)       128
_________________________________________________________________
activation_97 (Activation)   (None, 32, 200, 32)       0
_________________________________________________________________
dropout_38 (Dropout)         (None, 32, 200, 32)       0
_________________________________________________________________
up_sampling2d_19 (UpSampling (None, 64, 400, 32)       0
_________________________________________________________________
conv2d_101 (Conv2D)          (None, 64, 400, 32)       9248
_________________________________________________________________
batch_normalization_98 (Batc (None, 64, 400, 32)       128
_________________________________________________________________
activation_98 (Activation)   (None, 64, 400, 32)       0
_________________________________________________________________
conv2d_102 (Conv2D)          (None, 64, 400, 16)       4624
_________________________________________________________________
batch_normalization_99 (Batc (None, 64, 400, 16)       64
_________________________________________________________________
activation_99 (Activation)   (None, 64, 400, 16)       0
```

```
_________________________________________________________________
dropout_39 (Dropout)         (None, 64, 400, 16)       0
_________________________________________________________________
up_sampling2d_20 (UpSampling (None, 128, 800, 16)      0
_________________________________________________________________
conv2d_103 (Conv2D)          (None, 128, 800, 16)      2320
_________________________________________________________________
batch_normalization_100 (Bat (None, 128, 800, 16)      64
_________________________________________________________________
activation_100 (Activation)  (None, 128, 800, 16)      0
_________________________________________________________________
dropout_40 (Dropout)         (None, 128, 800, 16)      0
_________________________________________________________________
conv2d_104 (Conv2D)          (None, 128, 800, 4)       68
=================================================================
Total params: 1,849,012
Trainable params: 1,845,044
Non-trainable params: 3,968
_________________________________________________________________
```

2.1 Checkpointing the model and creating the callback list

In [0]:

```
tbc=TensorBoardColab()
mc = ModelCheckpoint('best_model_segnet.h5', monitor='val_dice_coef', verbose=1, save_b
est_only=True, mode='max')
callbacks_list = [mc, TensorBoardColabCallback(tbc)]
```

```
Wait for 8 seconds...
TensorBoard link:
http://513dcfc3.ngrok.io
```

2.2 Fitting the train data and validation

In [0]:

```
train_batches = Train_DataGenerator(train_data,shuffle=True)
valid_batches = Val_DataGenerator(cv_data)
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs =
30, verbose=1,
                              class_weight=class_wts, callbacks = callbacks_list)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Pleas
e use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please us
e tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/core.py:49: The name tf.summary.FileWriter is deprecated. Please use
tf.compat.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callb
acks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.c
ompat.v1.summary.merge_all instead.

Epoch 1/30
667/667 [==============================] - 263s 395ms/step - loss: 1.0959
- dice_coef: 0.0619 - val_loss: 1.1521 - val_dice_coef: 0.1376

Epoch 00001: val_dice_coef improved from -inf to 0.13762, saving model to
best_model.h5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.co
mpat.v1.Summary instead.

Epoch 2/30
667/667 [==============================] - 236s 353ms/step - loss: 0.7808
- dice_coef: 0.2758 - val_loss: 0.7062 - val_dice_coef: 0.3639

Epoch 00002: val_dice_coef improved from 0.13762 to 0.36390, saving model
to best_model.h5
Epoch 3/30
667/667 [==============================] - 238s 357ms/step - loss: 0.6896
- dice_coef: 0.3686 - val_loss: 0.6413 - val_dice_coef: 0.4156

Epoch 00003: val_dice_coef improved from 0.36390 to 0.41558, saving model
to best_model.h5
Epoch 4/30
667/667 [==============================] - 237s 356ms/step - loss: 0.6508
- dice_coef: 0.4057 - val_loss: 0.6436 - val_dice_coef: 0.4211

Epoch 00004: val_dice_coef improved from 0.41558 to 0.42105, saving model
to best_model.h5
Epoch 5/30
667/667 [==============================] - 238s 357ms/step - loss: 0.6210
- dice_coef: 0.4337 - val_loss: 0.6254 - val_dice_coef: 0.4399

Epoch 00005: val_dice_coef improved from 0.42105 to 0.43994, saving model
to best_model.h5
Epoch 6/30
667/667 [==============================] - 239s 358ms/step - loss: 0.6091
- dice_coef: 0.4445 - val_loss: 0.6371 - val_dice_coef: 0.4211

Epoch 00006: val_dice_coef did not improve from 0.43994
Epoch 7/30
667/667 [==============================] - 244s 366ms/step - loss: 0.5947
- dice_coef: 0.4569 - val_loss: 0.6352 - val_dice_coef: 0.4325

Epoch 00007: val_dice_coef did not improve from 0.43994
Epoch 8/30
```

```
667/667 [==============================] - 240s 360ms/step - loss: 0.5845
- dice_coef: 0.4672 - val_loss: 0.6019 - val_dice_coef: 0.4513

Epoch 00008: val_dice_coef improved from 0.43994 to 0.45134, saving model
to best_model.h5
Epoch 9/30
667/667 [==============================] - 239s 359ms/step - loss: 0.5736
- dice_coef: 0.4768 - val_loss: 0.5895 - val_dice_coef: 0.4672

Epoch 00009: val_dice_coef improved from 0.45134 to 0.46723, saving model
to best_model.h5
Epoch 10/30
667/667 [==============================] - 240s 360ms/step - loss: 0.5593
- dice_coef: 0.4895 - val_loss: 0.6522 - val_dice_coef: 0.4122

Epoch 00010: val_dice_coef did not improve from 0.46723
Epoch 11/30
667/667 [==============================] - 239s 358ms/step - loss: 0.5594
- dice_coef: 0.4894 - val_loss: 0.5801 - val_dice_coef: 0.4741

Epoch 00011: val_dice_coef improved from 0.46723 to 0.47410, saving model
to best_model.h5
Epoch 12/30
667/667 [==============================] - 240s 360ms/step - loss: 0.5619
- dice_coef: 0.4867 - val_loss: 0.6055 - val_dice_coef: 0.4546

Epoch 00012: val_dice_coef did not improve from 0.47410
Epoch 13/30
667/667 [==============================] - 241s 361ms/step - loss: 0.5493
- dice_coef: 0.4989 - val_loss: 0.5845 - val_dice_coef: 0.4650

Epoch 00013: val_dice_coef did not improve from 0.47410
Epoch 14/30
667/667 [==============================] - 240s 359ms/step - loss: 0.5404
- dice_coef: 0.5069 - val_loss: 0.5687 - val_dice_coef: 0.4856

Epoch 00014: val_dice_coef improved from 0.47410 to 0.48559, saving model
to best_model.h5
Epoch 15/30
667/667 [==============================] - 240s 359ms/step - loss: 0.5381
- dice_coef: 0.5094 - val_loss: 0.7156 - val_dice_coef: 0.3725

Epoch 00015: val_dice_coef did not improve from 0.48559
Epoch 16/30
667/667 [==============================] - 239s 359ms/step - loss: 0.5410
- dice_coef: 0.5058 - val_loss: 0.5595 - val_dice_coef: 0.4894

Epoch 00016: val_dice_coef improved from 0.48559 to 0.48944, saving model
to best_model.h5
Epoch 17/30
667/667 [==============================] - 240s 359ms/step - loss: 0.5360
- dice_coef: 0.5105 - val_loss: 0.5967 - val_dice_coef: 0.4563

Epoch 00017: val_dice_coef did not improve from 0.48944
Epoch 18/30
667/667 [==============================] - 239s 358ms/step - loss: 0.5295
- dice_coef: 0.5164 - val_loss: 0.5505 - val_dice_coef: 0.4981

Epoch 00018: val_dice_coef improved from 0.48944 to 0.49810, saving model
to best_model.h5
Epoch 19/30
```

```
667/667 [==============================] - 240s 359ms/step - loss: 0.5234
- dice_coef: 0.5230 - val_loss: 0.5758 - val_dice_coef: 0.4763

Epoch 00019: val_dice_coef did not improve from 0.49810
Epoch 20/30
667/667 [==============================] - 241s 361ms/step - loss: 0.5167
- dice_coef: 0.5284 - val_loss: 0.5626 - val_dice_coef: 0.4922

Epoch 00020: val_dice_coef did not improve from 0.49810
Epoch 21/30
667/667 [==============================] - 242s 363ms/step - loss: 0.5274
- dice_coef: 0.5178 - val_loss: 0.6572 - val_dice_coef: 0.4105

Epoch 00021: val_dice_coef did not improve from 0.49810
Epoch 22/30
667/667 [==============================] - 246s 368ms/step - loss: 0.5064
- dice_coef: 0.5377 - val_loss: 0.5872 - val_dice_coef: 0.4704

Epoch 00022: val_dice_coef did not improve from 0.49810
Epoch 23/30
667/667 [==============================] - 241s 361ms/step - loss: 0.5090
- dice_coef: 0.5348 - val_loss: 0.5943 - val_dice_coef: 0.4617

Epoch 00023: val_dice_coef did not improve from 0.49810
Epoch 24/30
667/667 [==============================] - 241s 361ms/step - loss: 0.5079
- dice_coef: 0.5363 - val_loss: 0.5492 - val_dice_coef: 0.5000

Epoch 00024: val_dice_coef improved from 0.49810 to 0.49997, saving model
to best_model.h5
Epoch 25/30
667/667 [==============================] - 240s 360ms/step - loss: 0.5093
- dice_coef: 0.5347 - val_loss: 0.6005 - val_dice_coef: 0.4578

Epoch 00025: val_dice_coef did not improve from 0.49997
Epoch 26/30
667/667 [==============================] - 239s 358ms/step - loss: 0.5058
- dice_coef: 0.5381 - val_loss: 0.5470 - val_dice_coef: 0.5024

Epoch 00026: val_dice_coef improved from 0.49997 to 0.50239, saving model
to best_model.h5
Epoch 27/30
667/667 [==============================] - 239s 358ms/step - loss: 0.5077
- dice_coef: 0.5356 - val_loss: 0.5569 - val_dice_coef: 0.4951

Epoch 00027: val_dice_coef did not improve from 0.50239
Epoch 28/30
667/667 [==============================] - 238s 356ms/step - loss: 0.4997
- dice_coef: 0.5436 - val_loss: 0.5490 - val_dice_coef: 0.4992

Epoch 00028: val_dice_coef did not improve from 0.50239
Epoch 29/30
667/667 [==============================] - 238s 356ms/step - loss: 0.4985
- dice_coef: 0.5445 - val_loss: 0.5404 - val_dice_coef: 0.5100

Epoch 00029: val_dice_coef improved from 0.50239 to 0.51004, saving model
to best_model.h5
Epoch 30/30
667/667 [==============================] - 236s 354ms/step - loss: 0.4886
- dice_coef: 0.5541 - val_loss: 0.5641 - val_dice_coef: 0.4931
```

```
Epoch 00030: val_dice_coef did not improve from 0.51004
```

Continued for another 20 epochs to see if the score improves

In [0]:

```
train_batches = Train_DataGenerator(train_data,shuffle=True)
valid_batches = Val_DataGenerator(cv_data)
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs =
20, verbose=1,
                              class_weight=class_wts, callbacks = callbacks_list)
```

```
Epoch 1/20
667/667 [==============================] - 246s 369ms/step - loss: 0.4945
- dice_coef: 0.5485 - val_loss: 0.5506 - val_dice_coef: 0.5031

Epoch 00001: val_dice_coef improved from -inf to 0.50306, saving model to
best_model.h5
Epoch 2/20
667/667 [==============================] - 240s 359ms/step - loss: 0.4942
- dice_coef: 0.5486 - val_loss: 0.5610 - val_dice_coef: 0.4920

Epoch 00002: val_dice_coef did not improve from 0.50306
Epoch 3/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4987
- dice_coef: 0.5444 - val_loss: 0.6268 - val_dice_coef: 0.4365

Epoch 00003: val_dice_coef did not improve from 0.50306
Epoch 4/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4955
- dice_coef: 0.5482 - val_loss: 0.5665 - val_dice_coef: 0.4886

Epoch 00004: val_dice_coef did not improve from 0.50306
Epoch 5/20
667/667 [==============================] - 240s 361ms/step - loss: 0.4876
- dice_coef: 0.5549 - val_loss: 0.5313 - val_dice_coef: 0.5168

Epoch 00005: val_dice_coef improved from 0.50306 to 0.51678, saving model
to best_model.h5
Epoch 6/20
667/667 [==============================] - 244s 365ms/step - loss: 0.4836
- dice_coef: 0.5584 - val_loss: 0.5563 - val_dice_coef: 0.4974

Epoch 00006: val_dice_coef did not improve from 0.51678
Epoch 7/20
667/667 [==============================] - 244s 365ms/step - loss: 0.4840
- dice_coef: 0.5582 - val_loss: 0.5821 - val_dice_coef: 0.4724

Epoch 00007: val_dice_coef did not improve from 0.51678
Epoch 8/20
667/667 [==============================] - 244s 365ms/step - loss: 0.4810
- dice_coef: 0.5614 - val_loss: 0.5663 - val_dice_coef: 0.4886

Epoch 00008: val_dice_coef did not improve from 0.51678
Epoch 9/20
667/667 [==============================] - 243s 364ms/step - loss: 0.4818
- dice_coef: 0.5605 - val_loss: 0.5494 - val_dice_coef: 0.5044

Epoch 00009: val_dice_coef did not improve from 0.51678
Epoch 10/20
667/667 [==============================] - 250s 374ms/step - loss: 0.4814
- dice_coef: 0.5605 - val_loss: 0.5572 - val_dice_coef: 0.4952

Epoch 00010: val_dice_coef did not improve from 0.51678
Epoch 11/20
667/667 [==============================] - 242s 363ms/step - loss: 0.4744
- dice_coef: 0.5668 - val_loss: 0.5548 - val_dice_coef: 0.4967

Epoch 00011: val_dice_coef did not improve from 0.51678
Epoch 12/20
667/667 [==============================] - 238s 357ms/step - loss: 0.4716
- dice_coef: 0.5698 - val_loss: 0.5408 - val_dice_coef: 0.5097
```

```
Epoch 00012: val_dice_coef did not improve from 0.51678
Epoch 13/20
667/667 [==============================] - 239s 358ms/step - loss: 0.4791
- dice_coef: 0.5629 - val_loss: 0.5572 - val_dice_coef: 0.4991

Epoch 00013: val_dice_coef did not improve from 0.51678
Epoch 14/20
667/667 [==============================] - 238s 357ms/step - loss: 0.4750
- dice_coef: 0.5665 - val_loss: 0.5366 - val_dice_coef: 0.5147

Epoch 00014: val_dice_coef did not improve from 0.51678
Epoch 15/20
667/667 [==============================] - 241s 361ms/step - loss: 0.4744
- dice_coef: 0.5667 - val_loss: 0.5767 - val_dice_coef: 0.4793

Epoch 00015: val_dice_coef did not improve from 0.51678
Epoch 16/20
667/667 [==============================] - 241s 361ms/step - loss: 0.4688
- dice_coef: 0.5727 - val_loss: 0.5620 - val_dice_coef: 0.4903

Epoch 00016: val_dice_coef did not improve from 0.51678
Epoch 17/20
667/667 [==============================] - 239s 358ms/step - loss: 0.4686
- dice_coef: 0.5734 - val_loss: 0.5236 - val_dice_coef: 0.5249

Epoch 00017: val_dice_coef improved from 0.51678 to 0.52488, saving model
to best_model.h5
Epoch 18/20
667/667 [==============================] - 238s 358ms/step - loss: 0.4636
- dice_coef: 0.5777 - val_loss: 0.5377 - val_dice_coef: 0.5160

Epoch 00018: val_dice_coef did not improve from 0.52488
Epoch 19/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4617
- dice_coef: 0.5794 - val_loss: 0.6208 - val_dice_coef: 0.4395

Epoch 00019: val_dice_coef did not improve from 0.52488
Epoch 20/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4670
- dice_coef: 0.5737 - val_loss: 0.5473 - val_dice_coef: 0.5049

Epoch 00020: val_dice_coef did not improve from 0.52488
```

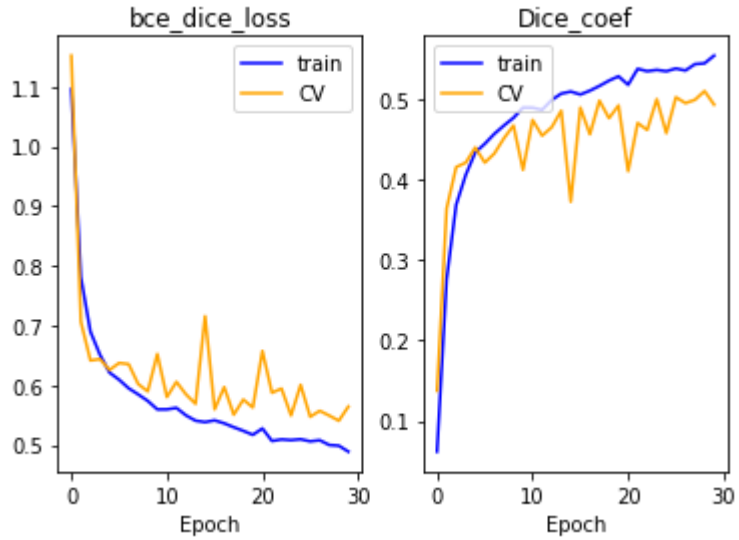
2.3 Plots on training & validation results

Loss function & metric plots

Please refer to Training Plots in utility functions section.

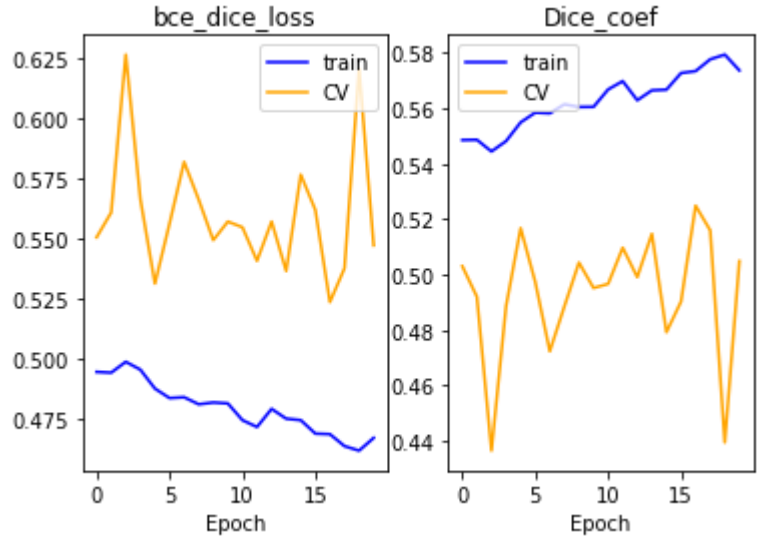
In [0]:

```
#first 30 epochs
plot(history)
```

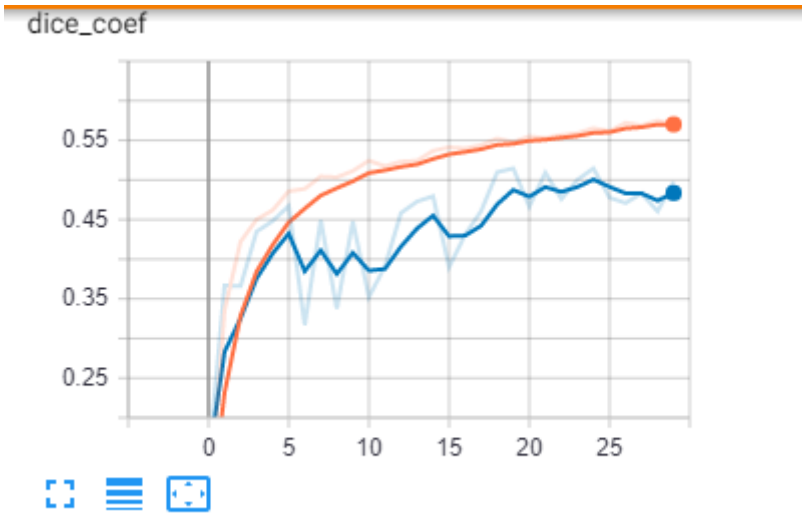


In [0]:

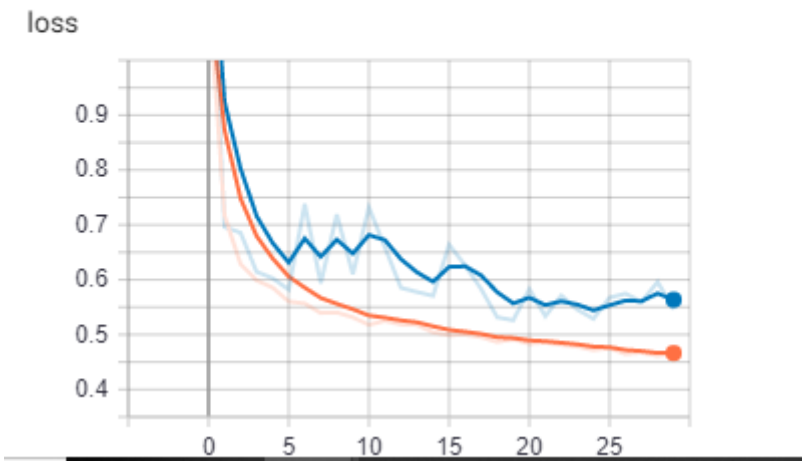
```
#next 20 epochs
plot(history)
```

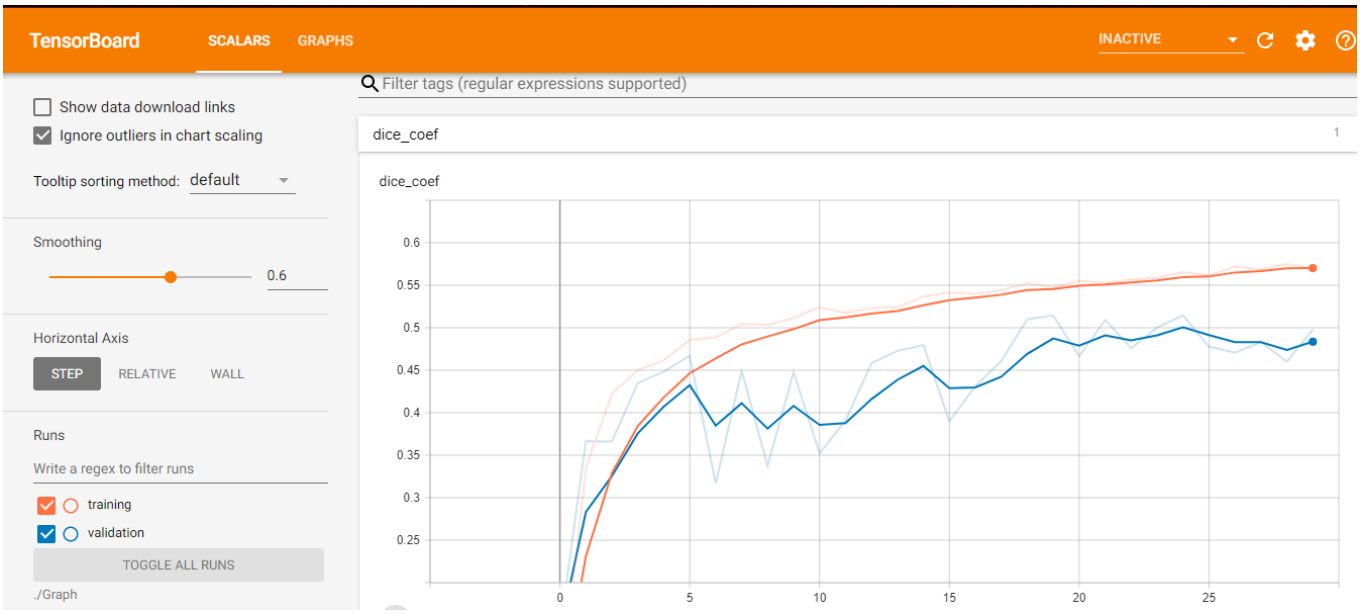


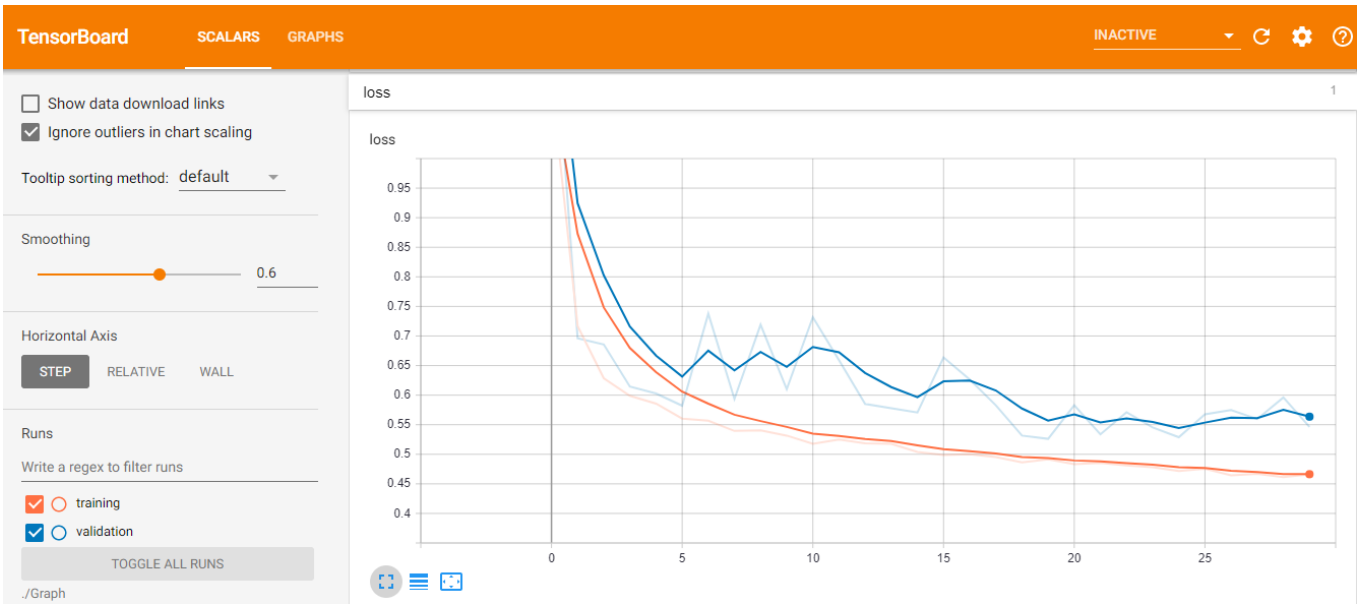
Tensorboard images


dice_coef
0.55
0.45
0.35
0.25
0
5
10
15
20
25

loss


loss
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25





2.4 Model Testing

Loading the best model for evaluation

Considering the best model which was trained on 30 epochs as the model in the next 20 epochs experienced the problem of overfitting.

In [0]:

```
from keras.models import load_model
dependencies = {'bce_dice_loss':bce_dice_loss,'dice_coef':dice_coef,}
model_best = load_model('/content/best_model_segnet.h5',custom_objects=dependencies)
```

Evaluating on validation images

In [0]:

```
evals= model_best.evaluate(valid_batches,verbose=1)
```

```
117/117 [==============================] - 31s 261ms/step
```

In [0]:

```
print('Validation set evaluation score:')
print('bce_dice loss:',evals[0])
print('dice_coeff:',evals[1])
```

```
Validation set evaluation score:
bce_dice loss: 0.5403748965925641
dice_coeff: 0.5100375587515171
```

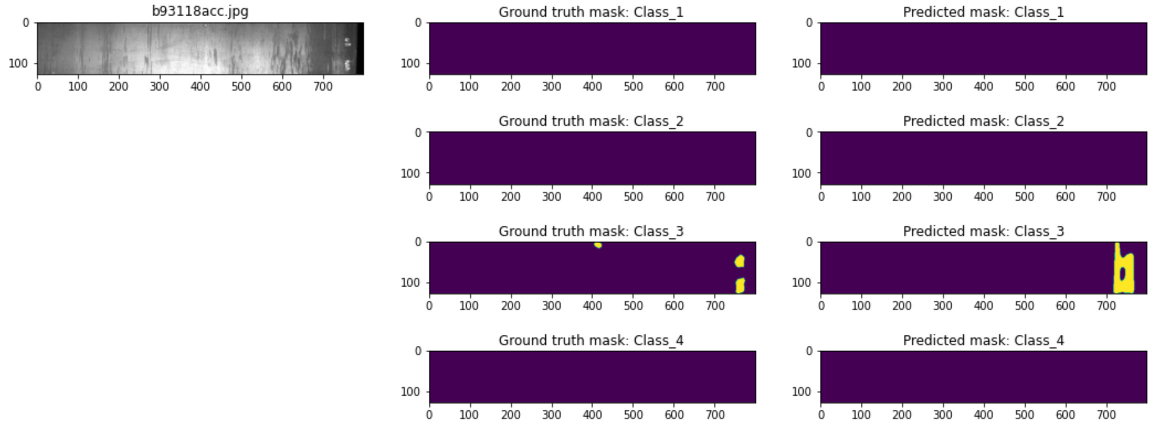
2.5 Defects visualization

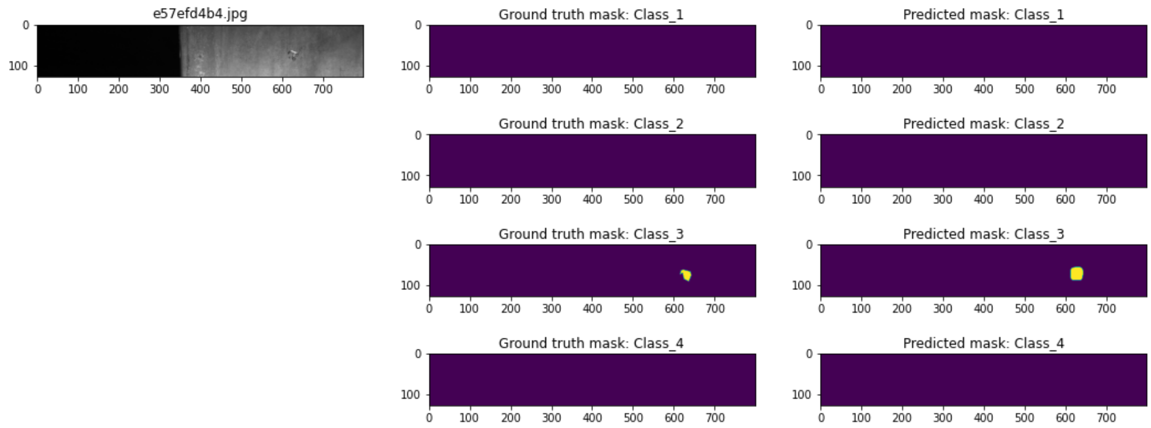
Please refer to the "visualize_defects" function in Utility functions section.

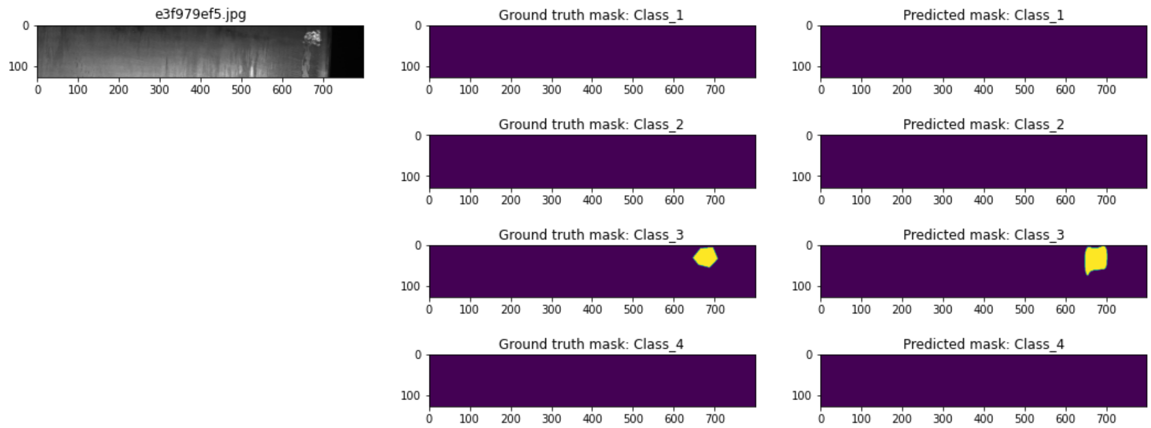
Training set

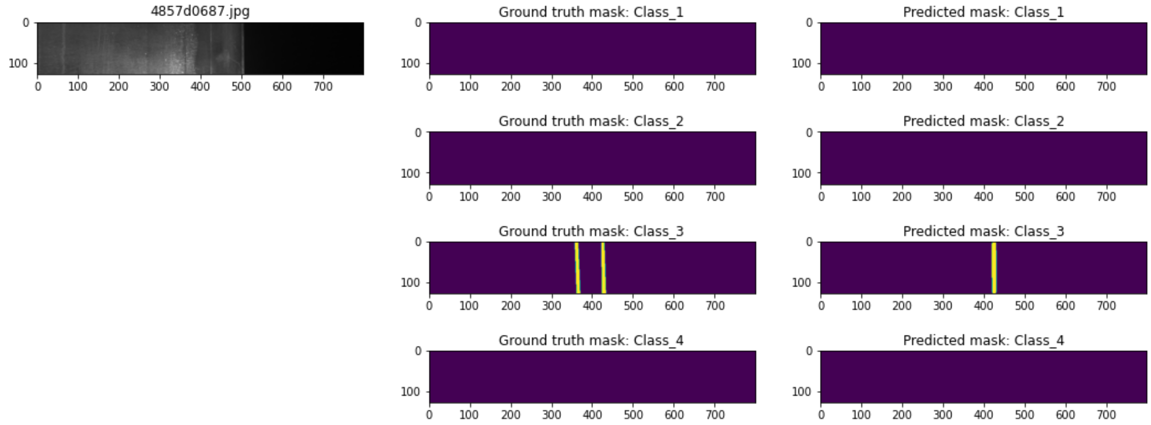
In [0]:

```
visualize_defects(train_data,model_best)
```

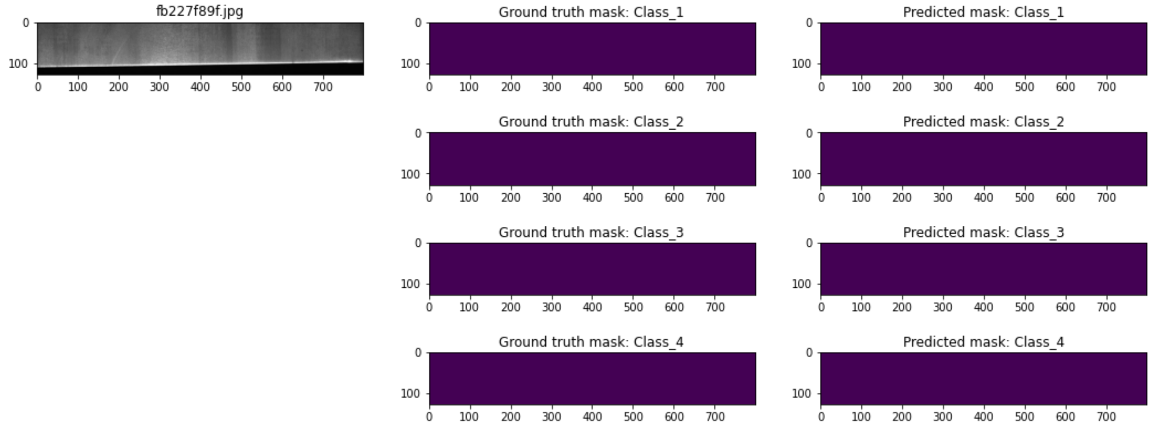

b93118acc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


e57efd4b4.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


e3f979ef5.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


4857d0687.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

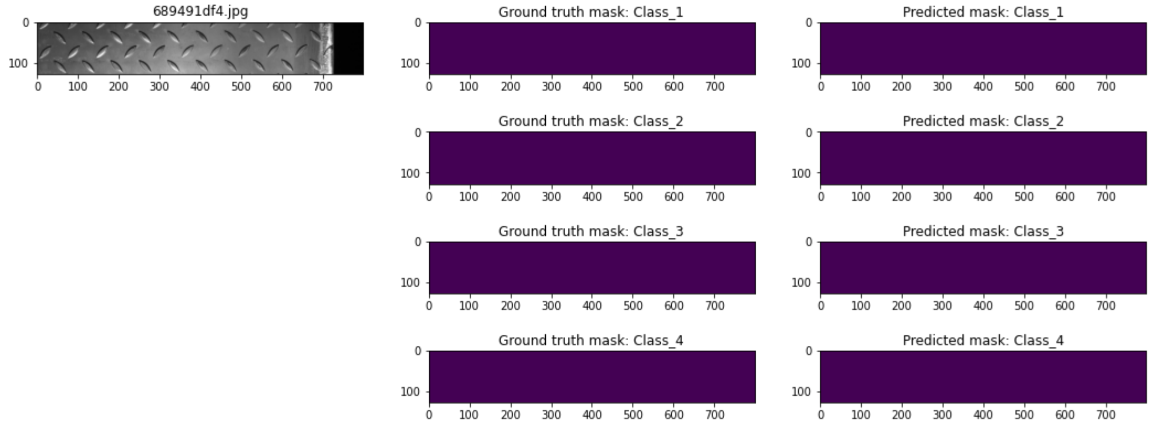
--

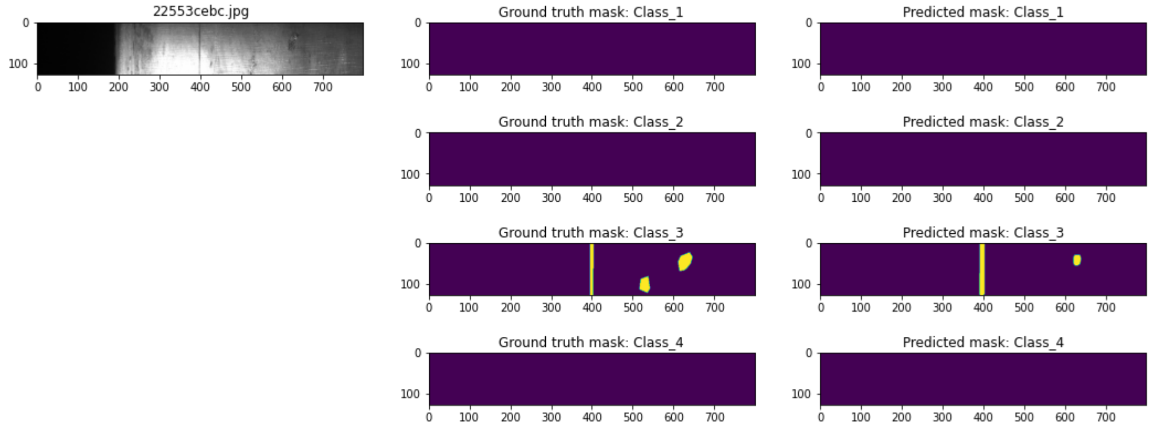

fb227f89f.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

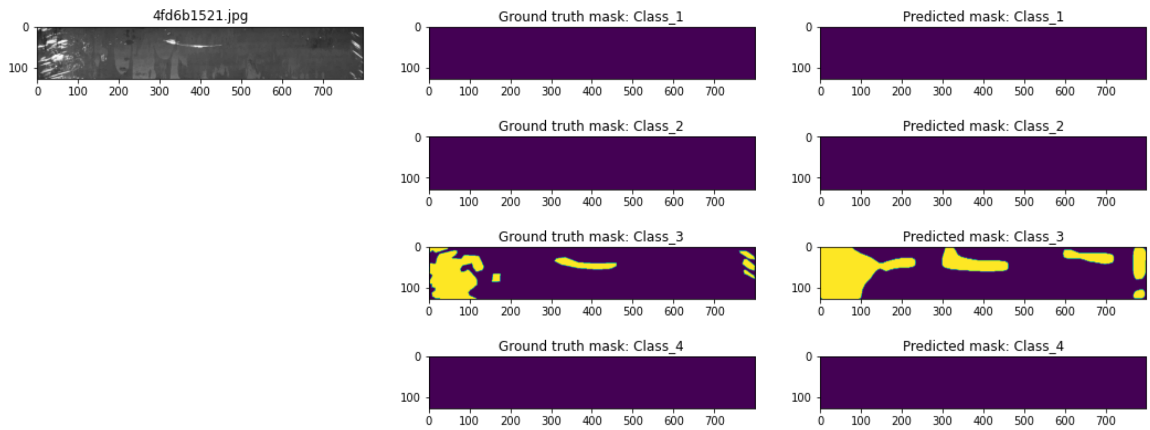
--

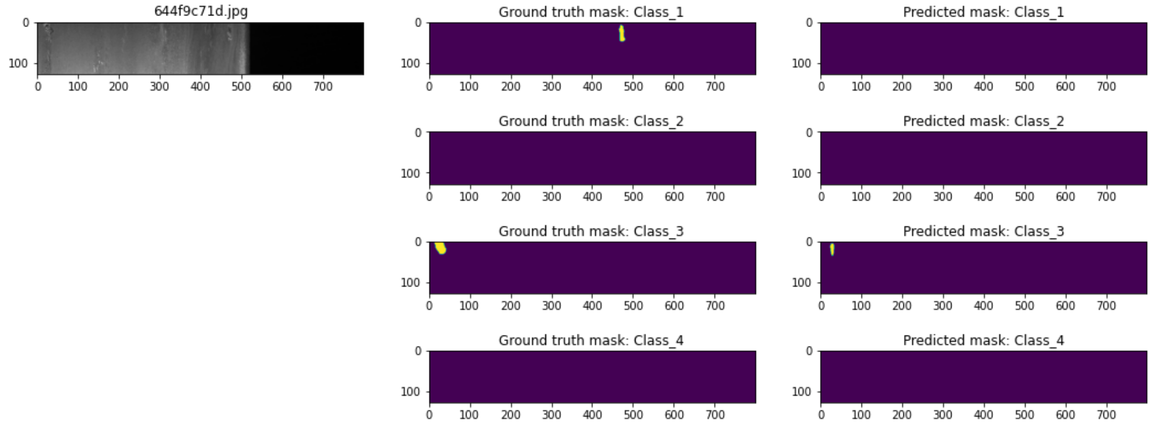
In [0]:

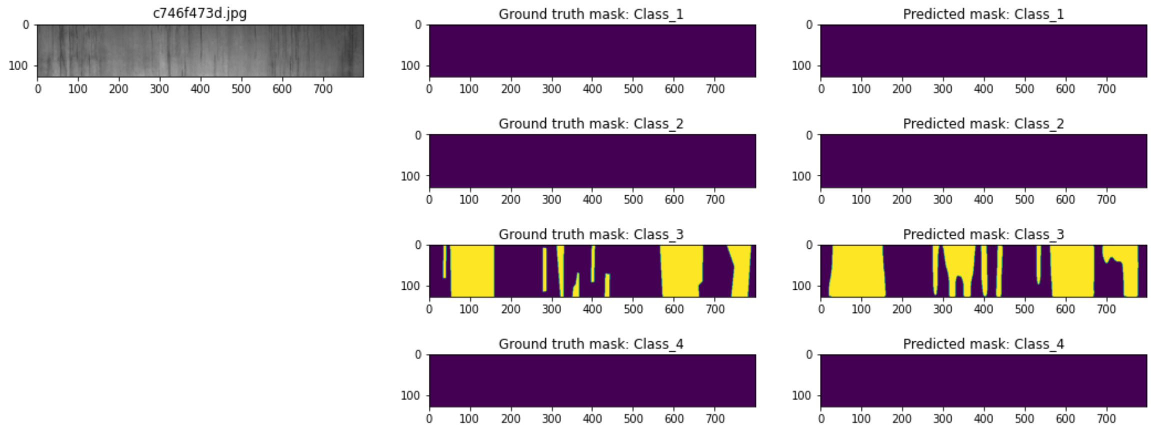
```
visualize_defects(train_data,model_best)
```


689491df4.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


22553cebc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


4fd6b1521.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

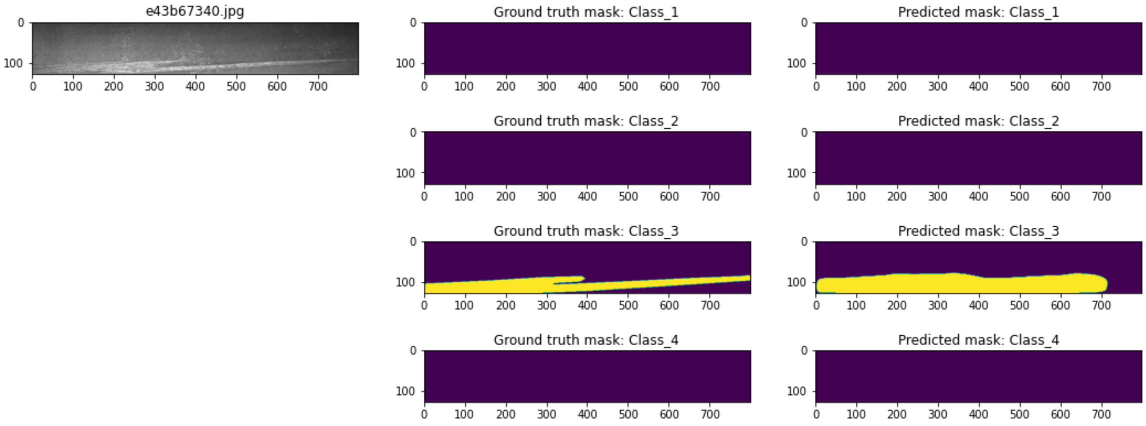


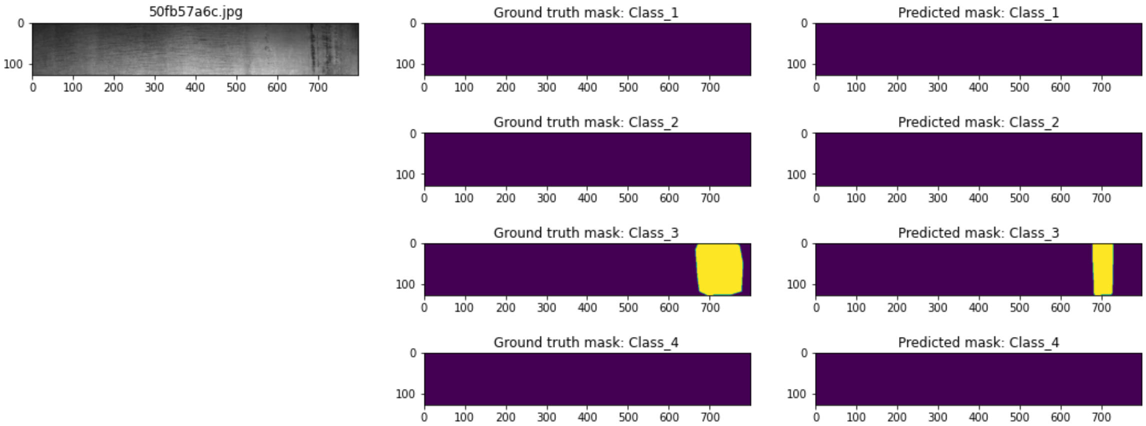


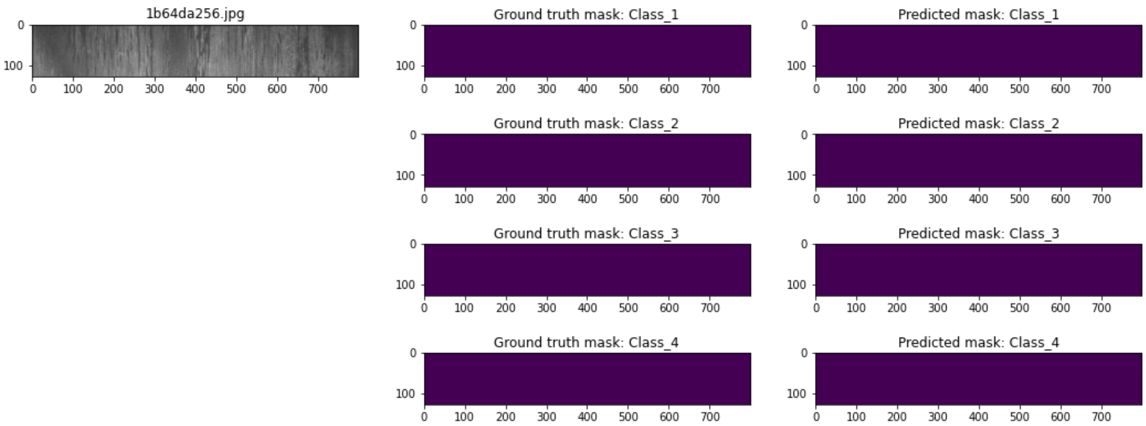
Validation set

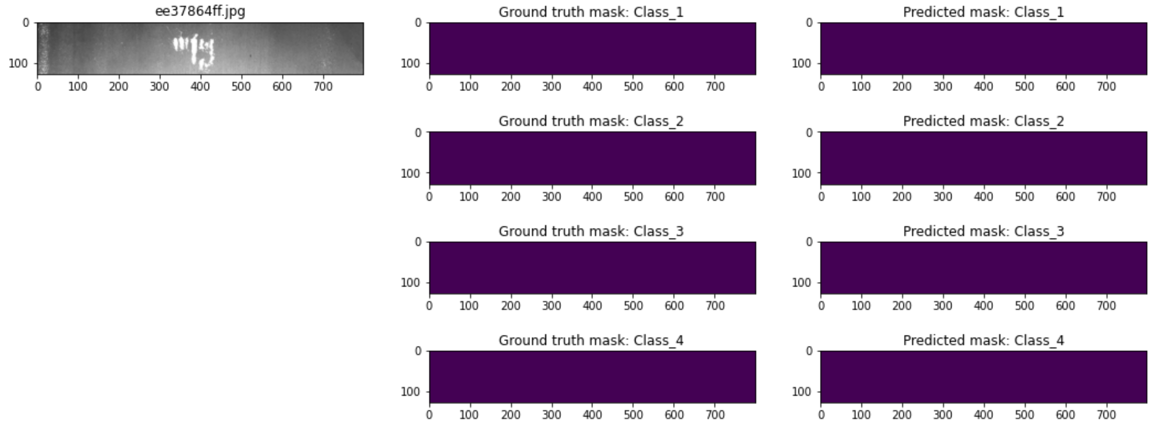
In [0]:

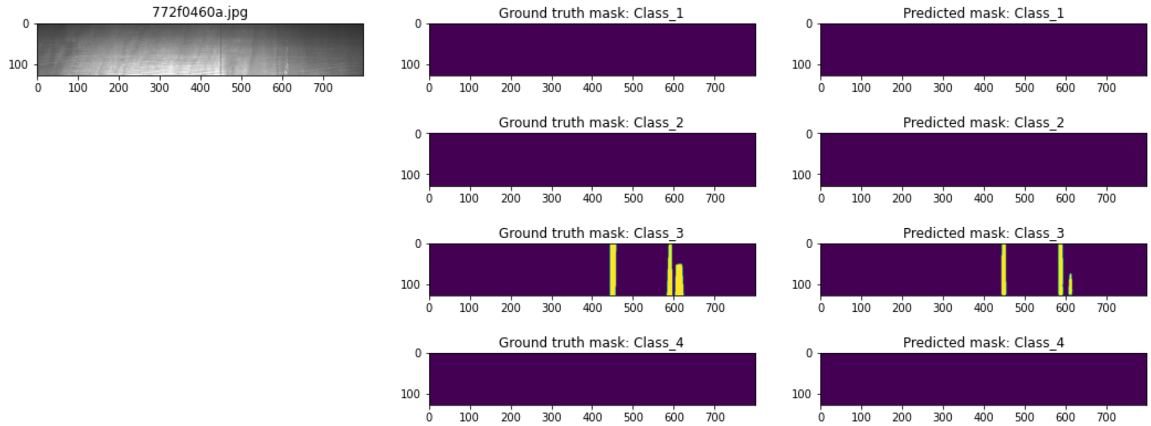
```
visualize_defects(cv_data,model_best)
```


e43b67340.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


50fb57a6c.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

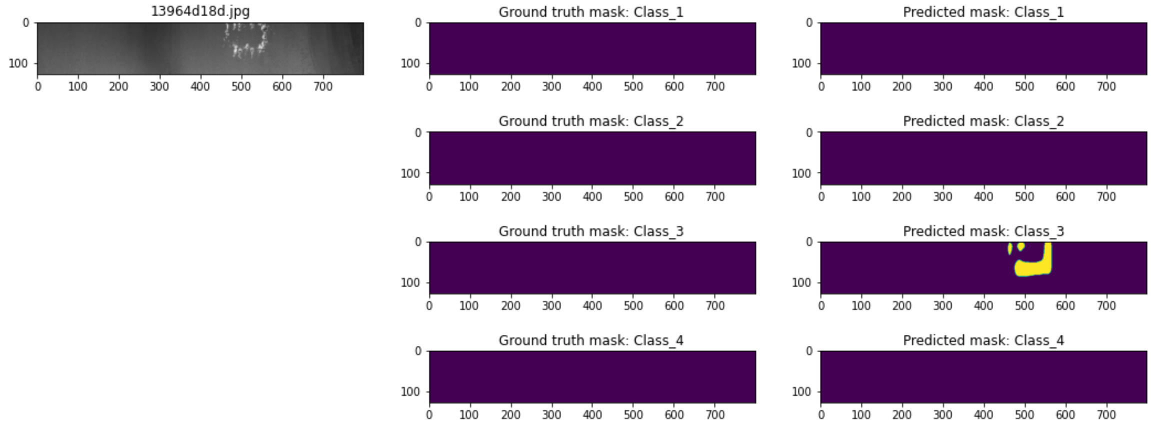

1b64da256.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


ee37864ff.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

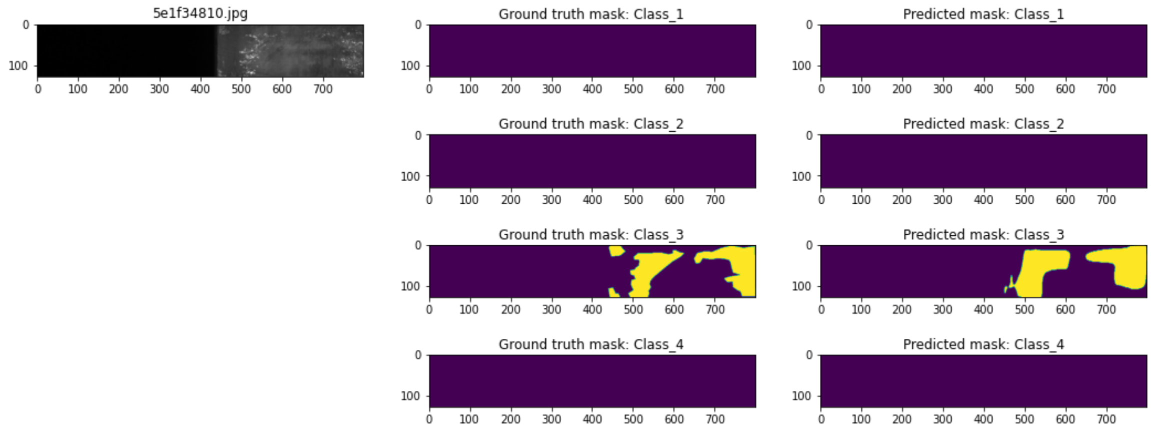

772f0460a.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

In [0]:

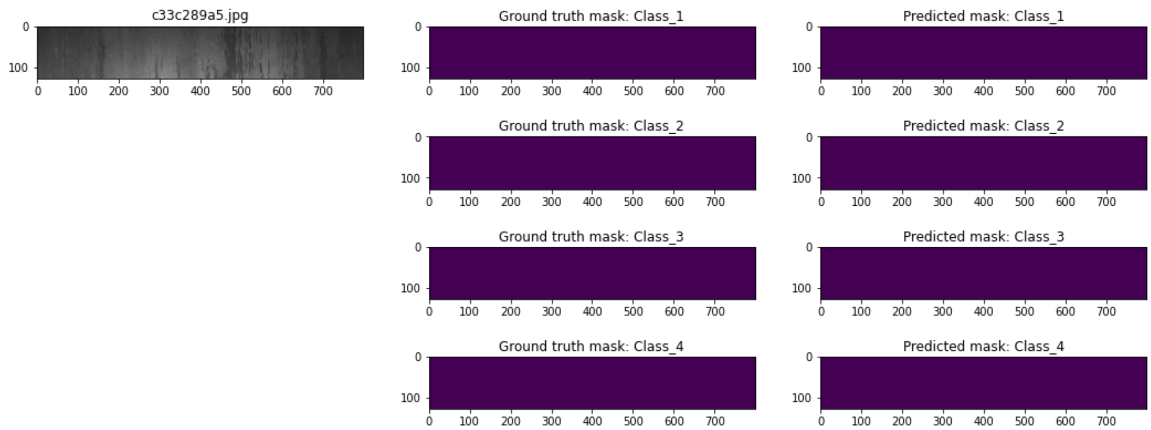
```
visualize_defects(cv_data,model_best)
```

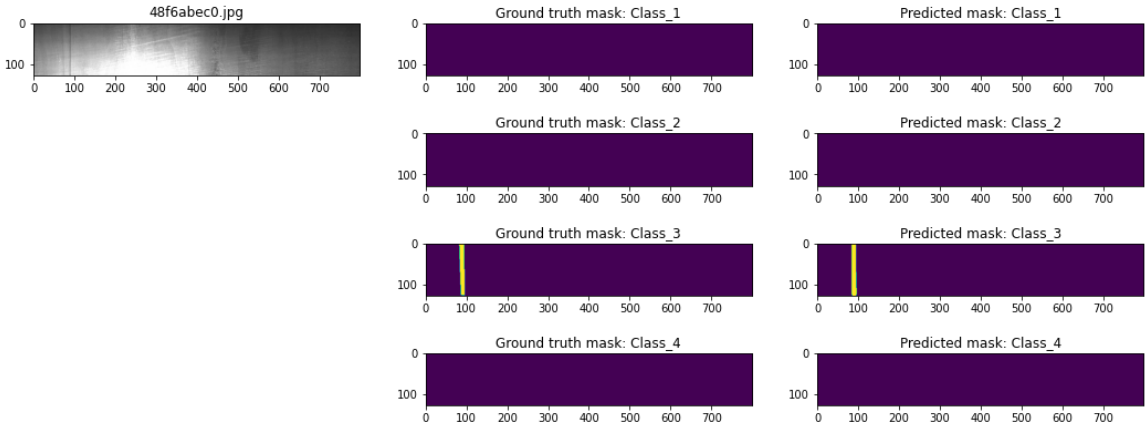

13964d18d.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

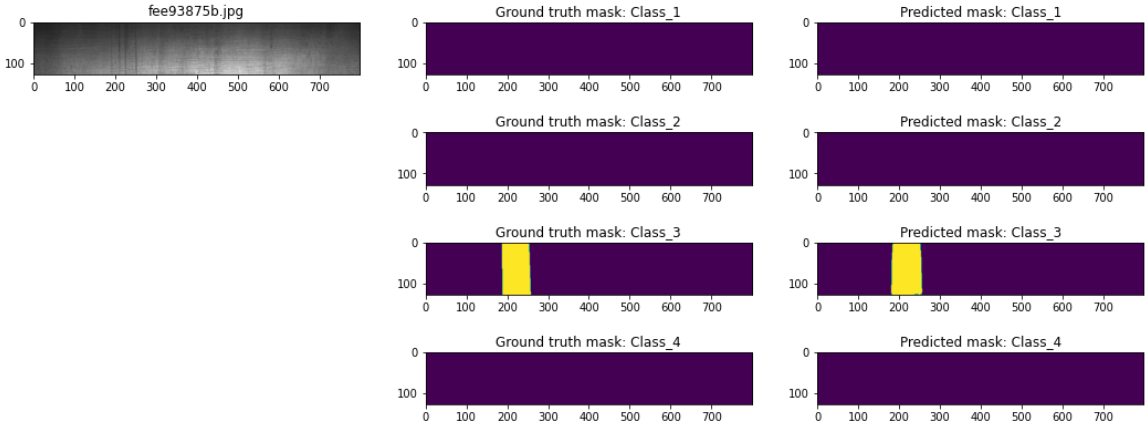
```
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--
```


5e1f34810.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

```
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--
```


c33c289a5.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4





2.6 Predicting defects on raw test images

Redefining the segnet architecture for original image size of 256x1600

In [0]:

```
input_img = Input((256,1600,3),name='img')
model1 = get_segnet(input_img, n_filters=16, dropout=0.5, batchnorm=True)
model1.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model1.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
img (InputLayer)             (None, 256, 1600, 3)      0
_________________________________________________________________
conv2d_105 (Conv2D)          (None, 256, 1600, 16)     448
_________________________________________________________________
batch_normalization_101 (Bat (None, 256, 1600, 16)     64
_________________________________________________________________
activation_101 (Activation)  (None, 256, 1600, 16)     0
_________________________________________________________________
conv2d_106 (Conv2D)          (None, 256, 1600, 16)     2320
_________________________________________________________________
batch_normalization_102 (Bat (None, 256, 1600, 16)     64
_________________________________________________________________
activation_102 (Activation)  (None, 256, 1600, 16)     0
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 128, 800, 16)      0
_________________________________________________________________
dropout_41 (Dropout)         (None, 128, 800, 16)      0
_________________________________________________________________
conv2d_107 (Conv2D)          (None, 128, 800, 32)      4640
_________________________________________________________________
batch_normalization_103 (Bat (None, 128, 800, 32)      128
_________________________________________________________________
activation_103 (Activation)  (None, 128, 800, 32)      0
_________________________________________________________________
conv2d_108 (Conv2D)          (None, 128, 800, 32)      9248
_________________________________________________________________
batch_normalization_104 (Bat (None, 128, 800, 32)      128
_________________________________________________________________
activation_104 (Activation)  (None, 128, 800, 32)      0
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 64, 400, 32)       0
_________________________________________________________________
dropout_42 (Dropout)         (None, 64, 400, 32)       0
_________________________________________________________________
conv2d_109 (Conv2D)          (None, 64, 400, 64)       18496
_________________________________________________________________
batch_normalization_105 (Bat (None, 64, 400, 64)       256
_________________________________________________________________
activation_105 (Activation)  (None, 64, 400, 64)       0
_________________________________________________________________
conv2d_110 (Conv2D)          (None, 64, 400, 64)       36928
_________________________________________________________________
batch_normalization_106 (Bat (None, 64, 400, 64)       256
_________________________________________________________________
activation_106 (Activation)  (None, 64, 400, 64)       0
_________________________________________________________________
conv2d_111 (Conv2D)          (None, 64, 400, 64)       36928
_________________________________________________________________
batch_normalization_107 (Bat (None, 64, 400, 64)       256
_________________________________________________________________
activation_107 (Activation)  (None, 64, 400, 64)       0
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 32, 200, 64)       0
_________________________________________________________________
dropout_43 (Dropout)         (None, 32, 200, 64)       0
_________________________________________________________________
conv2d_112 (Conv2D)          (None, 32, 200, 128)      73856
```

```
batch_normalization_108 (Bat (None, 32, 200, 128)      512
activation_108 (Activation)  (None, 32, 200, 128)      0
conv2d_113 (Conv2D)          (None, 32, 200, 128)      147584
batch_normalization_109 (Bat (None, 32, 200, 128)      512
activation_109 (Activation)  (None, 32, 200, 128)      0
conv2d_114 (Conv2D)          (None, 32, 200, 128)      147584
batch_normalization_110 (Bat (None, 32, 200, 128)      512
activation_110 (Activation)  (None, 32, 200, 128)      0
max_pooling2d_24 (MaxPooling (None, 16, 100, 128)      0
dropout_44 (Dropout)         (None, 16, 100, 128)      0
conv2d_115 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_111 (Bat (None, 16, 100, 128)      512
activation_111 (Activation)  (None, 16, 100, 128)      0
conv2d_116 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_112 (Bat (None, 16, 100, 128)      512
activation_112 (Activation)  (None, 16, 100, 128)      0
conv2d_117 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_113 (Bat (None, 16, 100, 128)      512
activation_113 (Activation)  (None, 16, 100, 128)      0
max_pooling2d_25 (MaxPooling (None, 8, 50, 128)        0
dropout_45 (Dropout)         (None, 8, 50, 128)        0
up_sampling2d_21 (UpSampling (None, 16, 100, 128)      0
conv2d_118 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_114 (Bat (None, 16, 100, 128)      512
activation_114 (Activation)  (None, 16, 100, 128)      0
conv2d_119 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_115 (Bat (None, 16, 100, 128)      512
activation_115 (Activation)  (None, 16, 100, 128)      0
conv2d_120 (Conv2D)          (None, 16, 100, 128)      147584
batch_normalization_116 (Bat (None, 16, 100, 128)      512
```

activation_116 (Activation)	(None, 16, 100, 128)	0
dropout_46 (Dropout)	(None, 16, 100, 128)	0
up_sampling2d_22 (UpSampling	(None, 32, 200, 128)	0
conv2d_121 (Conv2D)	(None, 32, 200, 128)	147584
batch_normalization_117 (Bat	(None, 32, 200, 128)	512
activation_117 (Activation)	(None, 32, 200, 128)	0
conv2d_122 (Conv2D)	(None, 32, 200, 128)	147584
batch_normalization_118 (Bat	(None, 32, 200, 128)	512
activation_118 (Activation)	(None, 32, 200, 128)	0
conv2d_123 (Conv2D)	(None, 32, 200, 64)	73792
batch_normalization_119 (Bat	(None, 32, 200, 64)	256
activation_119 (Activation)	(None, 32, 200, 64)	0
dropout_47 (Dropout)	(None, 32, 200, 64)	0
up_sampling2d_23 (UpSampling	(None, 64, 400, 64)	0
conv2d_124 (Conv2D)	(None, 64, 400, 64)	36928
batch_normalization_120 (Bat	(None, 64, 400, 64)	256
activation_120 (Activation)	(None, 64, 400, 64)	0
conv2d_125 (Conv2D)	(None, 64, 400, 64)	36928
batch_normalization_121 (Bat	(None, 64, 400, 64)	256
activation_121 (Activation)	(None, 64, 400, 64)	0
conv2d_126 (Conv2D)	(None, 64, 400, 32)	18464
batch_normalization_122 (Bat	(None, 64, 400, 32)	128
activation_122 (Activation)	(None, 64, 400, 32)	0
dropout_48 (Dropout)	(None, 64, 400, 32)	0
up_sampling2d_24 (UpSampling	(None, 128, 800, 32)	0
conv2d_127 (Conv2D)	(None, 128, 800, 32)	9248
batch_normalization_123 (Bat	(None, 128, 800, 32)	128
activation_123 (Activation)	(None, 128, 800, 32)	0
conv2d_128 (Conv2D)	(None, 128, 800, 16)	4624
batch_normalization_124 (Bat	(None, 128, 800, 16)	64
activation_124 (Activation)	(None, 128, 800, 16)	0

```
_________________________________________________________________
dropout_49 (Dropout)         (None, 128, 800, 16)      0
_________________________________________________________________
up_sampling2d_25 (UpSampling (None, 256, 1600, 16)     0
_________________________________________________________________
conv2d_129 (Conv2D)          (None, 256, 1600, 16)     2320
_________________________________________________________________
batch_normalization_125 (Bat (None, 256, 1600, 16)     64
_________________________________________________________________
activation_125 (Activation)  (None, 256, 1600, 16)     0
_________________________________________________________________
dropout_50 (Dropout)         (None, 256, 1600, 16)     0
_________________________________________________________________
conv2d_130 (Conv2D)          (None, 256, 1600, 4)      68
=================================================================
Total params: 1,849,012
Trainable params: 1,845,044
Non-trainable params: 3,968
_________________________________________________________________
```

In [0]:

```
model1.set_weights(model_best.get_weights())
```

Predicting on full 256x1600 raw test images

In [0]:

```
# Predicting on test images
from tqdm import tqdm
import cv2
data_path = '/content/' + 'test_images/'
files = list(os.listdir(data_path))
rle_lst = [] #list to store defect in run length encoding format
img_classId= [] #list to store Image ID + classId

for f in tqdm(files):
    X = np.empty((1,256,1600,3),dtype=np.float32)
    img = cv2.imread(data_path + f)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    X[0,] = img
    mask = model1.predict(X)
    rle_m = np.empty((256,1600),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))
```

```
100%|██████████| 5506/5506 [05:19<00:00, 17.21it/s]
```

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
output_df = pd.DataFrame(output)
output_df.to_csv('submission_segnet_256x1600.csv', index=False)
```

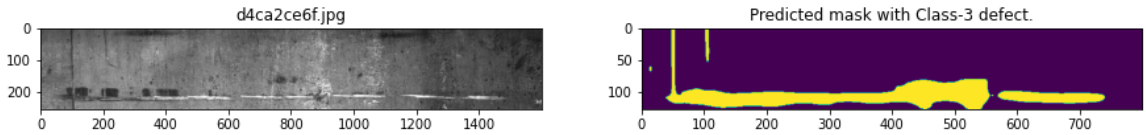
With this submission, I got a private dice coefficient score of 0.83437 & a public score of 0.84740.

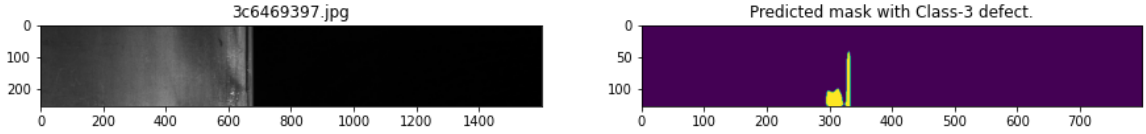
2.7 Visualizing defects of raw test images(256x1600)

Please refer to the "visualize_defects_test" function in Utility functions section.

In [0]:

```
visualize_defects(output_df,5)
```



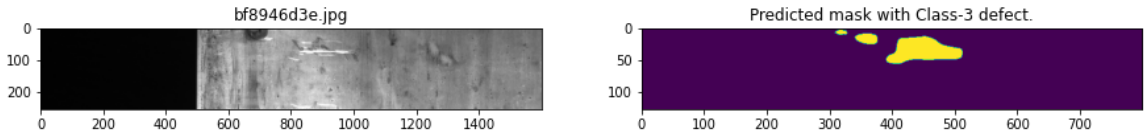


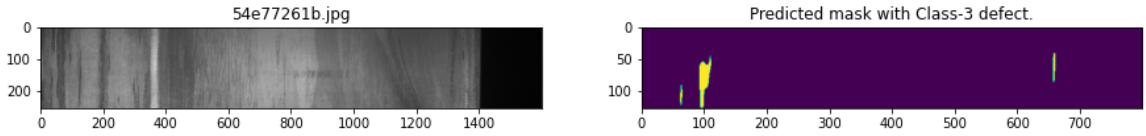
In [0]:

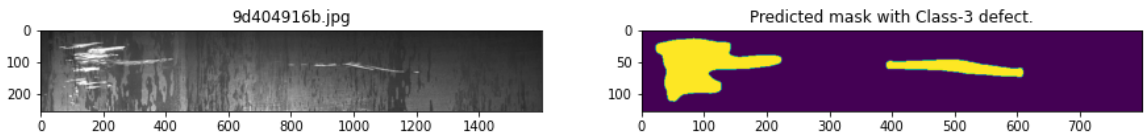
```
visualize_defects(output_df,5)
```

In [0]:

```
visualize_defects(output_df,5)
```

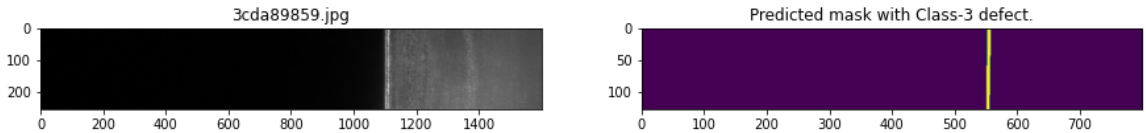






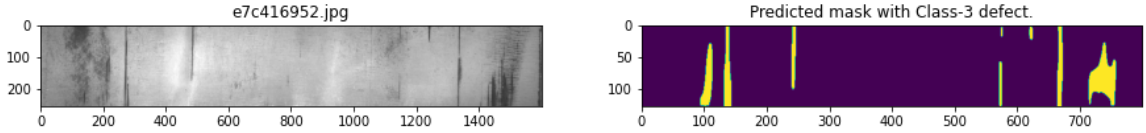
In [0]:

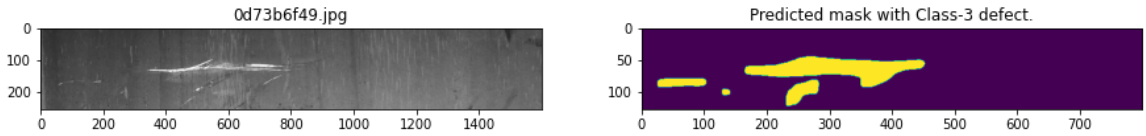
```
visualize_defects(output_df,5)
```

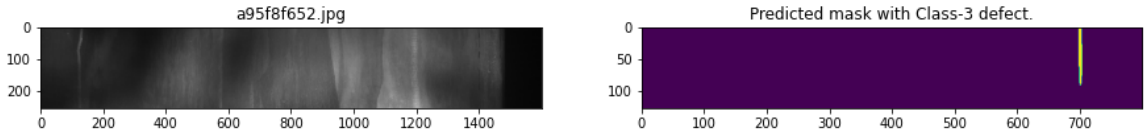


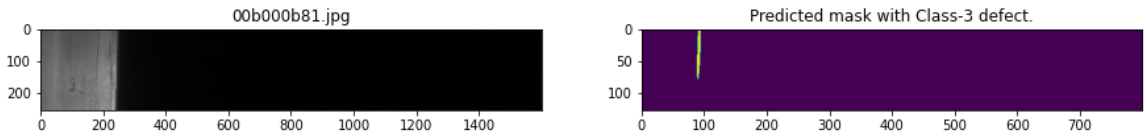
In [0]:

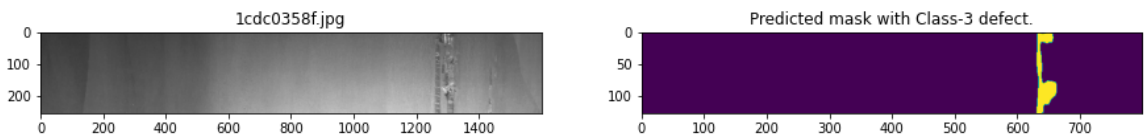
```
visualize_defects(output_df,5)
```



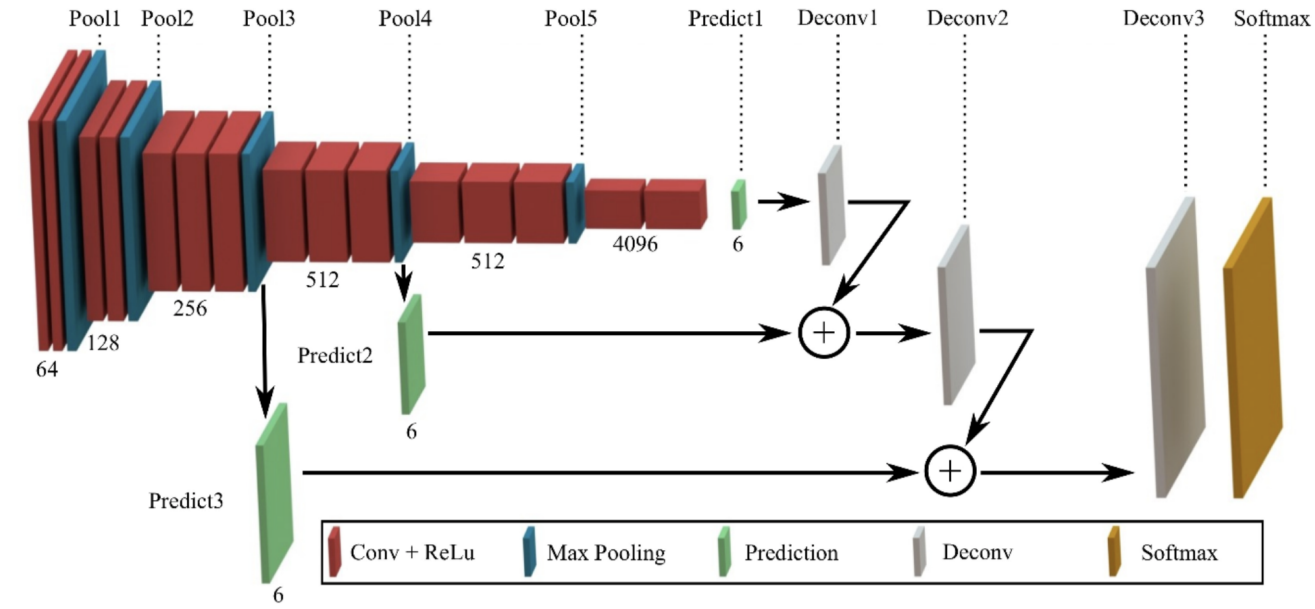




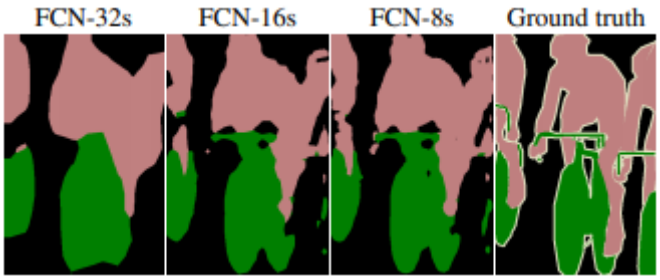




3.0 Implementing FCN-8 architecture



Original Skip Layer Architecture



References-

- https://github.com/seth814/Semantic-Shapes (https://github.com/seth814/Semantic-Shapes)
- https://fairyonice.github.io/Learn-about-Fully-Convolutional-Networks-for-semantic-segmentation.html (https://fairyonice.github.io/Learn-about-Fully-Convolutional-Networks-for-semantic-segmentation.html)
- https://www.youtube.com/watch?v=-3ylPH3BCWY (https://www.youtube.com/watch?v=-3ylPH3BCWY)

In [0]:

```
def fcn_8(img_shape, base, dropout,n_classes):
    i = Input(shape=img_shape)
    b= base

    ## Block 1
    x = Conv2D(2**b, (3, 3), activation='relu', kernel_initializer = 'he_normal', paddi
ng='same', name='block1_conv1')(i)
    x = BatchNormalization()(x)
    x = Conv2D(2**b, (3, 3), activation='relu', kernel_initializer = 'he_normal', paddi
ng='same', name='block1_conv2')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)
    x = Dropout(dropout)(x)

    # Block 2
    x = Conv2D(2**(b+1), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block2_conv1')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+1), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block2_conv2')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)
    x = Dropout(dropout)(x)

    # Block 3
    x = Conv2D(2**(b+2), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block3_conv1')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+2), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block3_conv2')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+2), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block3_conv3')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)
    x = Dropout(dropout)(x)
    pool3 = x

    # Block 4
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block4_conv1')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block4_conv2')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block4_conv3')(x)
    x = BatchNormalization()(x)
    pool4 = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)
    pool4 = Dropout(dropout)(pool4)

    # Block 5
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block5_conv1')(pool4)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
adding='same', name='block5_conv2')(x)
    x = BatchNormalization()(x)
    x = Conv2D(2**(b+3), (3, 3), activation='relu', kernel_initializer = 'he_normal', p
```

```
adding='same', name='block5_conv3')(x)
    x = BatchNormalization()(x)
    pool5 = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
    pool5 = Dropout(dropout)(pool5)

    conv6 = Conv2D(2**(b+3)*8 , (7, 7) , activation='relu', kernel_initializer = 'he_no
rmal' , padding='same', name="conv6")(pool5)
    conv6 = BatchNormalization()(conv6)
    conv6 = Dropout(0.5)(conv6)
    conv7 = Conv2D(2**(b+3)*8 , (1, 1) , activation='relu', kernel_initializer = 'he_no
rmal' , padding='same', name="conv7")(conv6)
    conv7 = BatchNormalization()(conv7)
    conv7 = Dropout(0.5)(conv7)

    pool4_n = Conv2D(n_classes, (1, 1), activation='relu', kernel_initializer = 'he_nor
mal', padding='same')(pool4)
    u2 = Conv2DTranspose(n_classes, kernel_size=(2, 2), strides=(2, 2), padding='same')
(conv7)
    u2_skip = Add()([pool4_n, u2])

    pool3_n = Conv2D(n_classes, (1, 1), activation='relu', padding='same')(pool3)
    u4 = Conv2DTranspose(n_classes, kernel_size=(2, 2), strides=(2, 2), padding='same')
(u2_skip)
    u4_skip = Add()([pool3_n, u4])

    output = Conv2DTranspose(n_classes, kernel_size=(8, 8), strides=(8, 8), padding='sa
me',activation='sigmoid')(u4_skip)

    model = Model(inputs=[i], outputs=[output])

    return model
```

In [0]:

```
#https://keras.io/models/model/#fit_generator
model= fcn_8(img_shape= (128,800,3), base= 4, dropout= 0.3, n_classes= 4)
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
input_2 (InputLayer)         (None, 128, 800, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 800, 16)      448
input_2[0][0]
_________________________________________________________________
batch_normalization_16 (BatchNo (None, 128, 800, 16)      64
block1_conv1[0][0]
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 800, 16)      2320
batch_normalization_16[0][0]
_________________________________________________________________
batch_normalization_17 (BatchNo (None, 128, 800, 16)      64
block1_conv2[0][0]
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 400, 16)       0
batch_normalization_17[0][0]
_________________________________________________________________
dropout_8 (Dropout)          (None, 64, 400, 16)       0
block1_pool[0][0]
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 400, 32)       4640
dropout_8[0][0]
_________________________________________________________________
batch_normalization_18 (BatchNo (None, 64, 400, 32)       128
block2_conv1[0][0]
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 400, 32)       9248
batch_normalization_18[0][0]
_________________________________________________________________
batch_normalization_19 (BatchNo (None, 64, 400, 32)       128
block2_conv2[0][0]
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 200, 32)       0
batch_normalization_19[0][0]
_________________________________________________________________
dropout_9 (Dropout)          (None, 32, 200, 32)       0
block2_pool[0][0]
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 200, 64)       18496
dropout_9[0][0]
_________________________________________________________________
```

```
_________________________
batch_normalization_20 (BatchNo (None, 32, 200, 64)  256         block3_co
nv1[0][0]
__________________________________________________________________________
_________________________
block3_conv2 (Conv2D)           (None, 32, 200, 64)  36928       batch_nor
malization_20[0][0]
__________________________________________________________________________
_________________________
batch_normalization_21 (BatchNo (None, 32, 200, 64)  256         block3_co
nv2[0][0]
__________________________________________________________________________
_________________________
block3_conv3 (Conv2D)           (None, 32, 200, 64)  36928       batch_nor
malization_21[0][0]
__________________________________________________________________________
_________________________
batch_normalization_22 (BatchNo (None, 32, 200, 64)  256         block3_co
nv3[0][0]
__________________________________________________________________________
_________________________
block3_pool (MaxPooling2D)      (None, 16, 100, 64)  0           batch_nor
malization_22[0][0]
__________________________________________________________________________
_________________________
dropout_10 (Dropout)            (None, 16, 100, 64)  0           block3_po
ol[0][0]
__________________________________________________________________________
_________________________
block4_conv1 (Conv2D)           (None, 16, 100, 128) 73856       dropout_1
0[0][0]
__________________________________________________________________________
_________________________
batch_normalization_23 (BatchNo (None, 16, 100, 128) 512         block4_co
nv1[0][0]
__________________________________________________________________________
_________________________
block4_conv2 (Conv2D)           (None, 16, 100, 128) 147584      batch_nor
malization_23[0][0]
__________________________________________________________________________
_________________________
batch_normalization_24 (BatchNo (None, 16, 100, 128) 512         block4_co
nv2[0][0]
__________________________________________________________________________
_________________________
block4_conv3 (Conv2D)           (None, 16, 100, 128) 147584      batch_nor
malization_24[0][0]
__________________________________________________________________________
_________________________
batch_normalization_25 (BatchNo (None, 16, 100, 128) 512         block4_co
nv3[0][0]
__________________________________________________________________________
_________________________
block4_pool (MaxPooling2D)      (None, 8, 50, 128)   0           batch_nor
malization_25[0][0]
__________________________________________________________________________
_________________________
dropout_11 (Dropout)            (None, 8, 50, 128)   0           block4_po
ol[0][0]
__________________________________________________________________________
_________________________
```

```
block5_conv1 (Conv2D)           (None, 8, 50, 128)   147584      dropout_1
1[0][0]
__________________________________________________________________________
_____________________
batch_normalization_26 (BatchNo (None, 8, 50, 128)   512         block5_co
nv1[0][0]
__________________________________________________________________________
_____________________
block5_conv2 (Conv2D)           (None, 8, 50, 128)   147584      batch_nor
malization_26[0][0]
__________________________________________________________________________
_____________________
batch_normalization_27 (BatchNo (None, 8, 50, 128)   512         block5_co
nv2[0][0]
__________________________________________________________________________
_____________________
block5_conv3 (Conv2D)           (None, 8, 50, 128)   147584      batch_nor
malization_27[0][0]
__________________________________________________________________________
_____________________
batch_normalization_28 (BatchNo (None, 8, 50, 128)   512         block5_co
nv3[0][0]
__________________________________________________________________________
_____________________
block5_pool (MaxPooling2D)      (None, 4, 25, 128)   0           batch_nor
malization_28[0][0]
__________________________________________________________________________
_____________________
dropout_12 (Dropout)            (None, 4, 25, 128)   0           block5_po
ol[0][0]
__________________________________________________________________________
_____________________
conv6 (Conv2D)                  (None, 4, 25, 1024)  6423552     dropout_1
2[0][0]
__________________________________________________________________________
_____________________
batch_normalization_29 (BatchNo (None, 4, 25, 1024)  4096        conv6[0]
[0]
__________________________________________________________________________
_____________________
dropout_13 (Dropout)            (None, 4, 25, 1024)  0           batch_nor
malization_29[0][0]
__________________________________________________________________________
_____________________
conv7 (Conv2D)                  (None, 4, 25, 1024)  1049600     dropout_1
3[0][0]
__________________________________________________________________________
_____________________
batch_normalization_30 (BatchNo (None, 4, 25, 1024)  4096        conv7[0]
[0]
__________________________________________________________________________
_____________________
dropout_14 (Dropout)            (None, 4, 25, 1024)  0           batch_nor
malization_30[0][0]
__________________________________________________________________________
_____________________
conv2d_3 (Conv2D)               (None, 8, 50, 4)     516         dropout_1
1[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_4 (Conv2DTrans (None, 8, 50, 4)     16388       dropout_1
```

```
4[0][0]
__________________________________________________________________________________
_____________________
add_3 (Add)                     (None, 8, 50, 4)     0           conv2d_3
[0][0]
                                                                 conv2d_tr
anspose_4[0][0]
__________________________________________________________________________________
_____________________
conv2d_4 (Conv2D)               (None, 16, 100, 4)   260         dropout_1
0[0][0]
__________________________________________________________________________________
_____________________
conv2d_transpose_5 (Conv2DTrans (None, 16, 100, 4)   68          add_3[0]
[0]
__________________________________________________________________________________
_____________________
add_4 (Add)                     (None, 16, 100, 4)   0           conv2d_4
[0][0]
                                                                 conv2d_tr
anspose_5[0][0]
__________________________________________________________________________________
_____________________
conv2d_transpose_6 (Conv2DTrans (None, 128, 800, 4)  1028        add_4[0]
[0]
==================================================================================
========================
Total params: 8,424,612
Trainable params: 8,418,404
Non-trainable params: 6,208
__________________________________________________________________________________
_____________________
```

3.1 Checkpointing the model and creating the callback list

In [0]:

```
tbc=TensorBoardColab()
mc = ModelCheckpoint('best_model_fcn8.h5', monitor='val_dice_coef', verbose=1, save_bes
t_only=True, mode='max')
callbacks_list = [mc, TensorBoardColabCallback(tbc)]
```

```
Wait for 8 seconds...
TensorBoard link:
https://358ac569.ngrok.io
```

3.2 Fitting the train data and validation

In [0]:

```
train_batches = Train_DataGenerator(train_data,shuffle=True)
valid_batches = Val_DataGenerator(cv_data)
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs =
30, verbose=1,
                              class_weight=class_wts, callbacks = callbacks_list)
```

```
ERROR! Session/line number was not unique in database. History logging mov
ed to new session 60
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Pleas
e use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please us
e tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/core.py:49: The name tf.summary.FileWriter is deprecated. Please use
tf.compat.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callb
acks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.c
ompat.v1.summary.merge_all instead.

Epoch 1/30
667/667 [==============================] - 262s 392ms/step - loss: 0.8073
- dice_coef: 0.2814 - val_loss: 0.7385 - val_dice_coef: 0.3199

Epoch 00001: val_dice_coef improved from -inf to 0.31990, saving model to
best_model.h5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.co
mpat.v1.Summary instead.

Epoch 2/30
667/667 [==============================] - 236s 354ms/step - loss: 0.6018
- dice_coef: 0.4443 - val_loss: 0.7356 - val_dice_coef: 0.3235

Epoch 00002: val_dice_coef improved from 0.31990 to 0.32351, saving model
to best_model.h5
Epoch 3/30
667/667 [==============================] - 238s 357ms/step - loss: 0.5655
- dice_coef: 0.4768 - val_loss: 0.6561 - val_dice_coef: 0.3929

Epoch 00003: val_dice_coef improved from 0.32351 to 0.39290, saving model
to best_model.h5
Epoch 4/30
667/667 [==============================] - 239s 358ms/step - loss: 0.5419
- dice_coef: 0.4991 - val_loss: 0.5570 - val_dice_coef: 0.4868

Epoch 00004: val_dice_coef improved from 0.39290 to 0.48675, saving model
to best_model.h5
Epoch 5/30
667/667 [==============================] - 237s 355ms/step - loss: 0.5287
- dice_coef: 0.5118 - val_loss: 0.8987 - val_dice_coef: 0.1731

Epoch 00005: val_dice_coef did not improve from 0.48675
Epoch 6/30
667/667 [==============================] - 236s 354ms/step - loss: 0.5213
- dice_coef: 0.5176 - val_loss: 0.5272 - val_dice_coef: 0.5125

Epoch 00006: val_dice_coef improved from 0.48675 to 0.51246, saving model
to best_model.h5
Epoch 7/30
667/667 [==============================] - 237s 355ms/step - loss: 0.5106
- dice_coef: 0.5279 - val_loss: 0.5975 - val_dice_coef: 0.4473
```

```
Epoch 00007: val_dice_coef did not improve from 0.51246
Epoch 8/30
667/667 [==============================] - 237s 356ms/step - loss: 0.5019
- dice_coef: 0.5359 - val_loss: 0.5189 - val_dice_coef: 0.5206

Epoch 00008: val_dice_coef improved from 0.51246 to 0.52060, saving model
to best_model.h5
Epoch 9/30
667/667 [==============================] - 239s 358ms/step - loss: 0.4876
- dice_coef: 0.5491 - val_loss: 0.6301 - val_dice_coef: 0.4224

Epoch 00009: val_dice_coef did not improve from 0.52060
Epoch 10/30
667/667 [==============================] - 239s 358ms/step - loss: 0.4790
- dice_coef: 0.5578 - val_loss: 0.5311 - val_dice_coef: 0.5095

Epoch 00010: val_dice_coef did not improve from 0.52060
Epoch 11/30
667/667 [==============================] - 238s 357ms/step - loss: 0.4614
- dice_coef: 0.5740 - val_loss: 0.5877 - val_dice_coef: 0.4614

Epoch 00011: val_dice_coef did not improve from 0.52060
Epoch 12/30
667/667 [==============================] - 238s 357ms/step - loss: 0.4479
- dice_coef: 0.5862 - val_loss: 1.0221 - val_dice_coef: 0.0805

Epoch 00012: val_dice_coef did not improve from 0.52060
Epoch 13/30
667/667 [==============================] - 238s 357ms/step - loss: 0.4405
- dice_coef: 0.5940 - val_loss: 0.5188 - val_dice_coef: 0.5230

Epoch 00013: val_dice_coef improved from 0.52060 to 0.52296, saving model
to best_model.h5
Epoch 14/30
667/667 [==============================] - 238s 357ms/step - loss: 0.4296
- dice_coef: 0.6049 - val_loss: 0.6698 - val_dice_coef: 0.3900

Epoch 00014: val_dice_coef did not improve from 0.52296
Epoch 15/30
667/667 [==============================] - 239s 358ms/step - loss: 0.4174
- dice_coef: 0.6153 - val_loss: 0.5601 - val_dice_coef: 0.4845

Epoch 00015: val_dice_coef did not improve from 0.52296
Epoch 16/30
667/667 [==============================] - 238s 357ms/step - loss: 0.4189
- dice_coef: 0.6137 - val_loss: 0.5328 - val_dice_coef: 0.5116

Epoch 00016: val_dice_coef did not improve from 0.52296
Epoch 17/30
667/667 [==============================] - 239s 359ms/step - loss: 0.4013
- dice_coef: 0.6302 - val_loss: 0.5416 - val_dice_coef: 0.5084

Epoch 00017: val_dice_coef did not improve from 0.52296
Epoch 18/30
667/667 [==============================] - 239s 358ms/step - loss: 0.4004
- dice_coef: 0.6315 - val_loss: 0.5761 - val_dice_coef: 0.4751

Epoch 00018: val_dice_coef did not improve from 0.52296
Epoch 19/30
667/667 [==============================] - 240s 361ms/step - loss: 0.3974
- dice_coef: 0.6340 - val_loss: 0.8097 - val_dice_coef: 0.2653
```

```
Epoch 00019: val_dice_coef did not improve from 0.52296
Epoch 20/30
667/667 [==============================] - 239s 359ms/step - loss: 0.3906
- dice_coef: 0.6410 - val_loss: 0.5715 - val_dice_coef: 0.4794

Epoch 00020: val_dice_coef did not improve from 0.52296
Epoch 21/30
667/667 [==============================] - 241s 361ms/step - loss: 0.3869
- dice_coef: 0.6440 - val_loss: 0.5087 - val_dice_coef: 0.5357

Epoch 00021: val_dice_coef improved from 0.52296 to 0.53567, saving model
to best_model.h5
Epoch 22/30
667/667 [==============================] - 240s 360ms/step - loss: 0.3887
- dice_coef: 0.6425 - val_loss: 0.6491 - val_dice_coef: 0.4090

Epoch 00022: val_dice_coef did not improve from 0.53567
Epoch 23/30
667/667 [==============================] - 241s 362ms/step - loss: 0.3774
- dice_coef: 0.6527 - val_loss: 0.5835 - val_dice_coef: 0.4682

Epoch 00023: val_dice_coef did not improve from 0.53567
Epoch 24/30
667/667 [==============================] - 240s 360ms/step - loss: 0.3769
- dice_coef: 0.6533 - val_loss: 0.5461 - val_dice_coef: 0.5020

Epoch 00024: val_dice_coef did not improve from 0.53567
Epoch 25/30
667/667 [==============================] - 242s 363ms/step - loss: 0.3630
- dice_coef: 0.6658 - val_loss: 0.5320 - val_dice_coef: 0.5144

Epoch 00025: val_dice_coef did not improve from 0.53567
Epoch 26/30
667/667 [==============================] - 245s 367ms/step - loss: 0.3636
- dice_coef: 0.6660 - val_loss: 0.5159 - val_dice_coef: 0.5279

Epoch 00026: val_dice_coef did not improve from 0.53567
Epoch 27/30
667/667 [==============================] - 246s 369ms/step - loss: 0.3597
- dice_coef: 0.6687 - val_loss: 0.7179 - val_dice_coef: 0.3530

Epoch 00027: val_dice_coef did not improve from 0.53567
Epoch 28/30
667/667 [==============================] - 242s 363ms/step - loss: 0.3620
- dice_coef: 0.6665 - val_loss: 0.5897 - val_dice_coef: 0.4616

Epoch 00028: val_dice_coef did not improve from 0.53567
Epoch 29/30
667/667 [==============================] - 242s 363ms/step - loss: 0.3553
- dice_coef: 0.6726 - val_loss: 0.5552 - val_dice_coef: 0.4953

Epoch 00029: val_dice_coef did not improve from 0.53567
Epoch 30/30
667/667 [==============================] - 240s 360ms/step - loss: 0.3535
- dice_coef: 0.6746 - val_loss: 0.5141 - val_dice_coef: 0.5302

Epoch 00030: val_dice_coef did not improve from 0.53567
```

In [0]:

```
#round 2 for 20 epochs
model= model_best
train_batches = Train_DataGenerator(train_data,shuffle=True)
valid_batches = Val_DataGenerator(cv_data)
class_wts= [1.98,7.18,0.34,2.21]
history = model.fit_generator(train_batches, validation_data = valid_batches, epochs =
20, verbose=1,
                              class_weight=class_wts, callbacks = callbacks_list)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/core.py:49: The name tf.summary.FileWriter is deprecated. Please use
tf.compat.v1.summary.FileWriter instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callb
acks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.c
ompat.v1.summary.merge_all instead.

Epoch 1/20
667/667 [==============================] - 260s 389ms/step - loss: 0.5078
- dice_coef: 0.5295 - val_loss: 0.5507 - val_dice_coef: 0.5024

Epoch 00001: val_dice_coef improved from -inf to 0.50236, saving model to
best_model.h5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorboard
colab/callbacks.py:51: The name tf.Summary is deprecated. Please use tf.co
mpat.v1.Summary instead.

Epoch 2/20
667/667 [==============================] - 244s 366ms/step - loss: 0.5027
- dice_coef: 0.5344 - val_loss: 0.5386 - val_dice_coef: 0.5093

Epoch 00002: val_dice_coef improved from 0.50236 to 0.50929, saving model
to best_model.h5
Epoch 3/20
667/667 [==============================] - 251s 376ms/step - loss: 0.4952
- dice_coef: 0.5414 - val_loss: 0.5888 - val_dice_coef: 0.4639

Epoch 00003: val_dice_coef did not improve from 0.50929
Epoch 4/20
667/667 [==============================] - 249s 374ms/step - loss: 0.4775
- dice_coef: 0.5588 - val_loss: 0.5142 - val_dice_coef: 0.5323

Epoch 00004: val_dice_coef improved from 0.50929 to 0.53226, saving model
to best_model.h5
Epoch 5/20
667/667 [==============================] - 241s 362ms/step - loss: 0.4601
- dice_coef: 0.5747 - val_loss: 0.6080 - val_dice_coef: 0.4484

Epoch 00005: val_dice_coef did not improve from 0.53226
Epoch 6/20
667/667 [==============================] - 241s 362ms/step - loss: 0.4497
- dice_coef: 0.5850 - val_loss: 0.7672 - val_dice_coef: 0.3144

Epoch 00006: val_dice_coef did not improve from 0.53226
Epoch 7/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4421
- dice_coef: 0.5913 - val_loss: 0.5534 - val_dice_coef: 0.5002

Epoch 00007: val_dice_coef did not improve from 0.53226
Epoch 8/20
667/667 [==============================] - 240s 360ms/step - loss: 0.4405
- dice_coef: 0.5932 - val_loss: 0.6820 - val_dice_coef: 0.3835

Epoch 00008: val_dice_coef did not improve from 0.53226
Epoch 9/20
667/667 [==============================] - 239s 359ms/step - loss: 0.4238
- dice_coef: 0.6085 - val_loss: 0.6206 - val_dice_coef: 0.4392

Epoch 00009: val_dice_coef did not improve from 0.53226
Epoch 10/20
```

```
667/667 [==============================] - 242s 363ms/step - loss: 0.4134
- dice_coef: 0.6189 - val_loss: 0.5577 - val_dice_coef: 0.4948

Epoch 00010: val_dice_coef did not improve from 0.53226
Epoch 11/20
667/667 [==============================] - 241s 362ms/step - loss: 0.4039
- dice_coef: 0.6272 - val_loss: 0.6911 - val_dice_coef: 0.3801

Epoch 00011: val_dice_coef did not improve from 0.53226
Epoch 12/20
667/667 [==============================] - 244s 366ms/step - loss: 0.4045
- dice_coef: 0.6274 - val_loss: 0.5428 - val_dice_coef: 0.5117

Epoch 00012: val_dice_coef did not improve from 0.53226
Epoch 13/20
667/667 [==============================] - 246s 369ms/step - loss: 0.4046
- dice_coef: 0.6270 - val_loss: 0.5021 - val_dice_coef: 0.5462

Epoch 00013: val_dice_coef improved from 0.53226 to 0.54618, saving model
to best_model.h5
Epoch 14/20
667/667 [==============================] - 250s 375ms/step - loss: 0.3877
- dice_coef: 0.6427 - val_loss: 0.4743 - val_dice_coef: 0.5711

Epoch 00014: val_dice_coef improved from 0.54618 to 0.57108, saving model
to best_model.h5
Epoch 15/20
667/667 [==============================] - 241s 361ms/step - loss: 0.3896
- dice_coef: 0.6407 - val_loss: 0.9046 - val_dice_coef: 0.2007

Epoch 00015: val_dice_coef did not improve from 0.57108
Epoch 16/20
667/667 [==============================] - 246s 368ms/step - loss: 0.3856
- dice_coef: 0.6449 - val_loss: 0.4810 - val_dice_coef: 0.5668

Epoch 00016: val_dice_coef did not improve from 0.57108
Epoch 17/20
667/667 [==============================] - 247s 371ms/step - loss: 0.3815
- dice_coef: 0.6487 - val_loss: 0.6442 - val_dice_coef: 0.4214

Epoch 00017: val_dice_coef did not improve from 0.57108
Epoch 18/20
667/667 [==============================] - 249s 373ms/step - loss: 0.3723
- dice_coef: 0.6573 - val_loss: 0.6303 - val_dice_coef: 0.4345

Epoch 00018: val_dice_coef did not improve from 0.57108
Epoch 19/20
667/667 [==============================] - 249s 373ms/step - loss: 0.3709
- dice_coef: 0.6579 - val_loss: 0.5422 - val_dice_coef: 0.5118

Epoch 00019: val_dice_coef did not improve from 0.57108
Epoch 20/20
667/667 [==============================] - 251s 376ms/step - loss: 0.3637
- dice_coef: 0.6647 - val_loss: 0.6284 - val_dice_coef: 0.4359

Epoch 00020: val_dice_coef did not improve from 0.57108
```

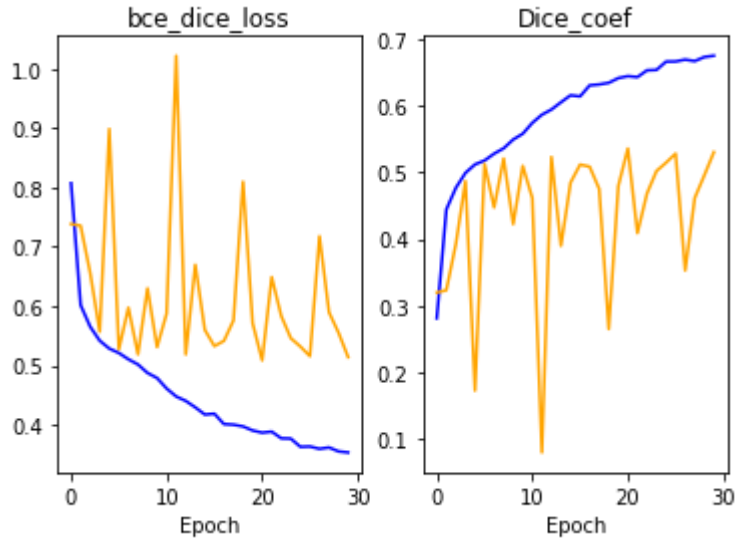
3.3 Plots on training & validation results

Loss function & metric plots

Please refer to Training Plots in utility functions section.

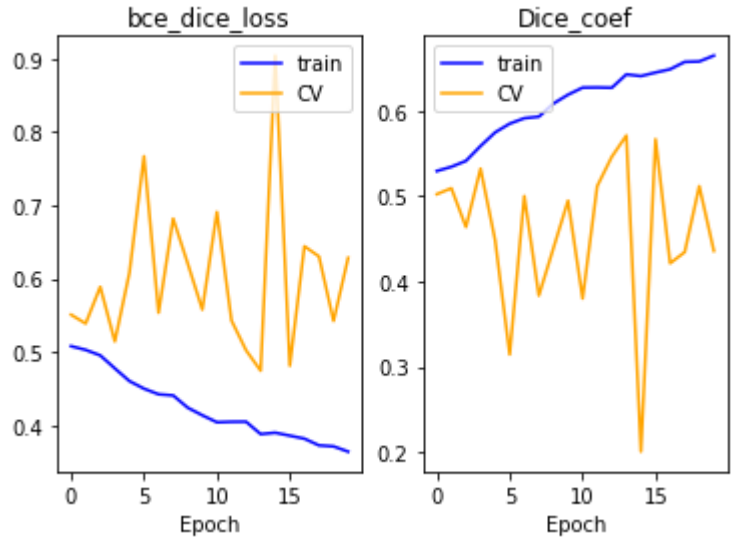
In [0]:

```
#first 30 epochs
plot(history)
```

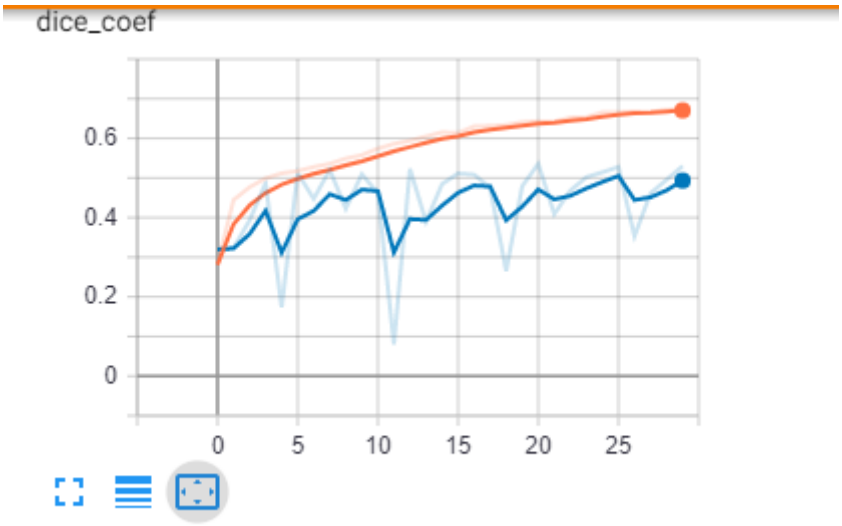


In [0]:

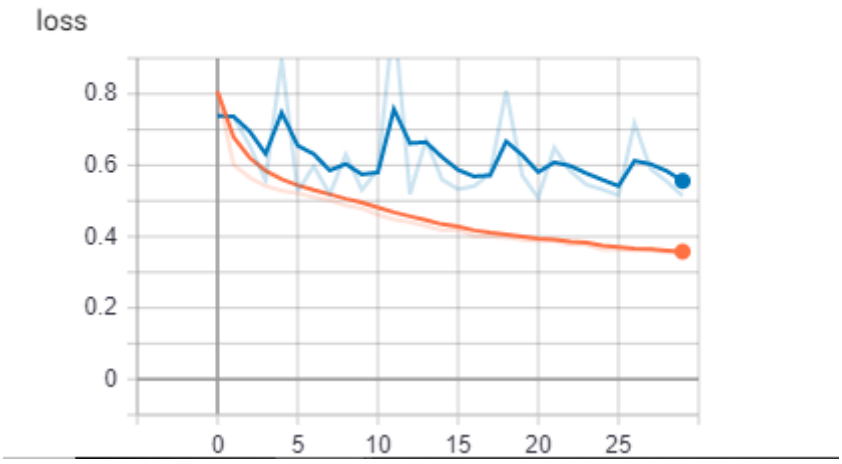
```
#next 20 epochs
plot(history)
```



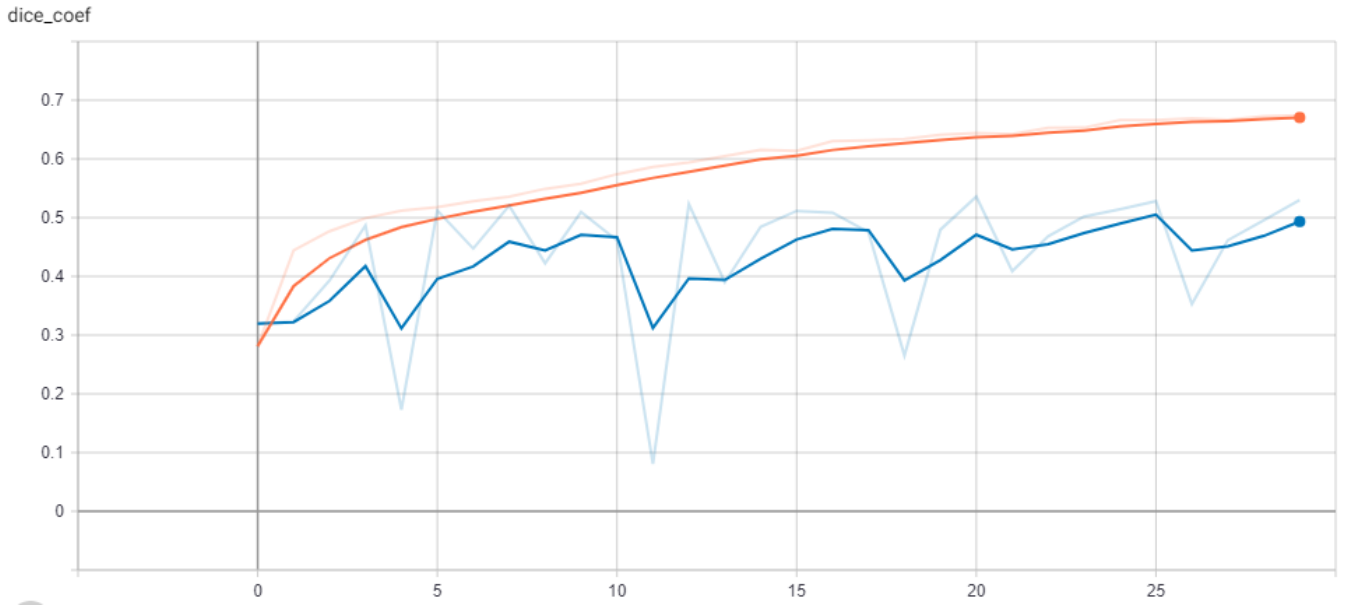
Tensorboard images

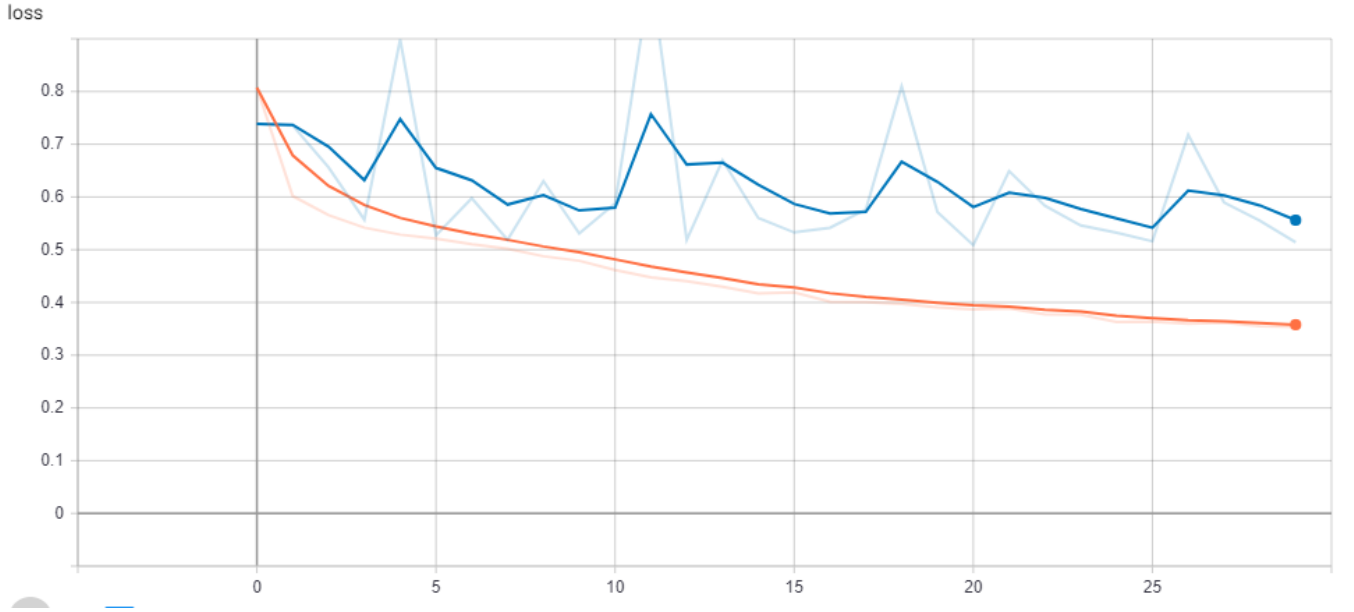


loss



dice_coef 1


dice_coef
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25


loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25

3.4 Model Testing

Loading the best model for evaluation

Considering the best model which was trained on 30 epochs as the model in the next 20 epochs experienced the problem of overfitting.

In [0]:

```
from keras.models import load_model
dependencies = {'bce_dice_loss':bce_dice_loss,'dice_coef':dice_coef,}
model_best = load_model('/content/best_model_fcn8.h5',custom_objects=dependencies)
```

Evaluating on validation images

In [0]:

```
evals1= model_best.evaluate(valid_batches,verbose=1)
```

```
117/117 [==============================] - 33s 282ms/step
```

In [0]:

```
print('Validation set evaluation score:')
print('bce_dice loss:',evals1[0])
print('dice_coeff:',evals1[1])
```

```
Validation set evaluation score:
bce_dice loss: 0.5122087106236026
dice_coeff: 0.531455701447896
```

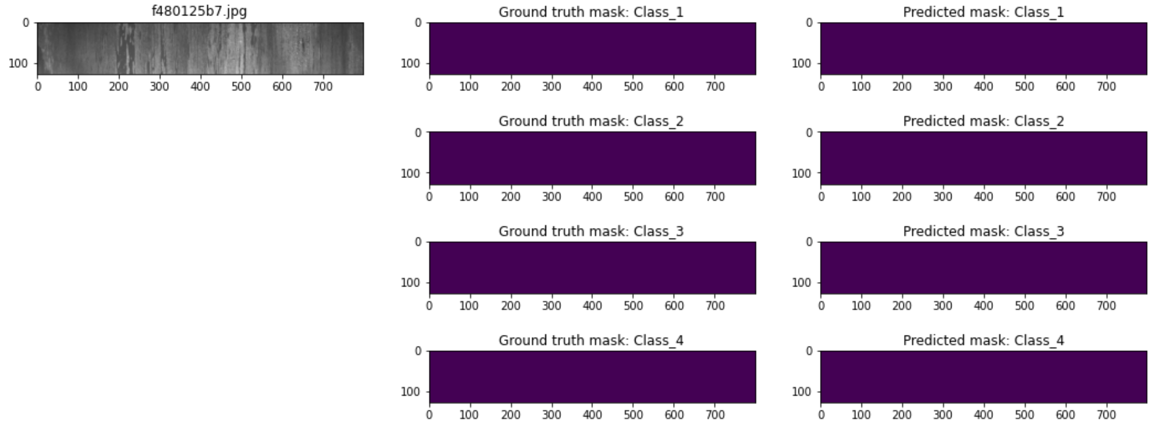
3.5 Defects visualization

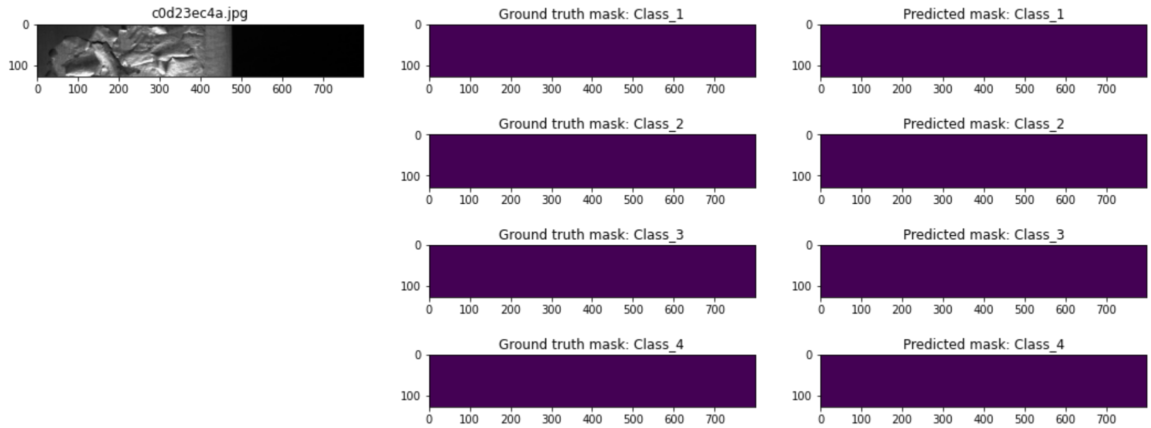
Please refer to the "visualize_defects" function in Utility functions section.

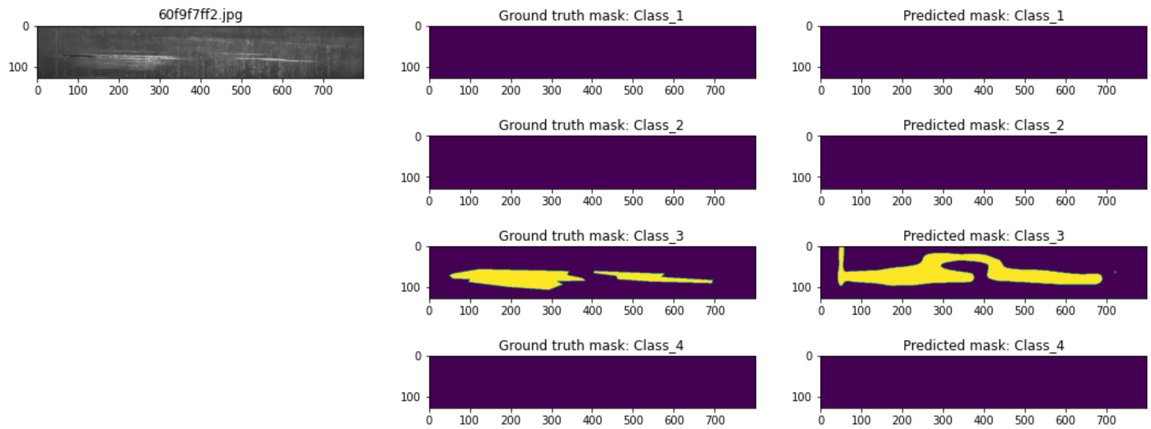
Training set

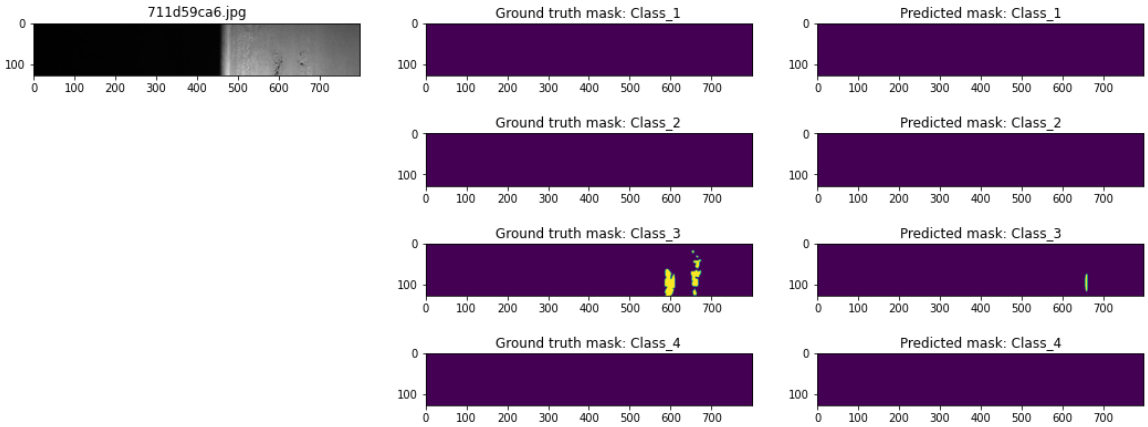
In [0]:

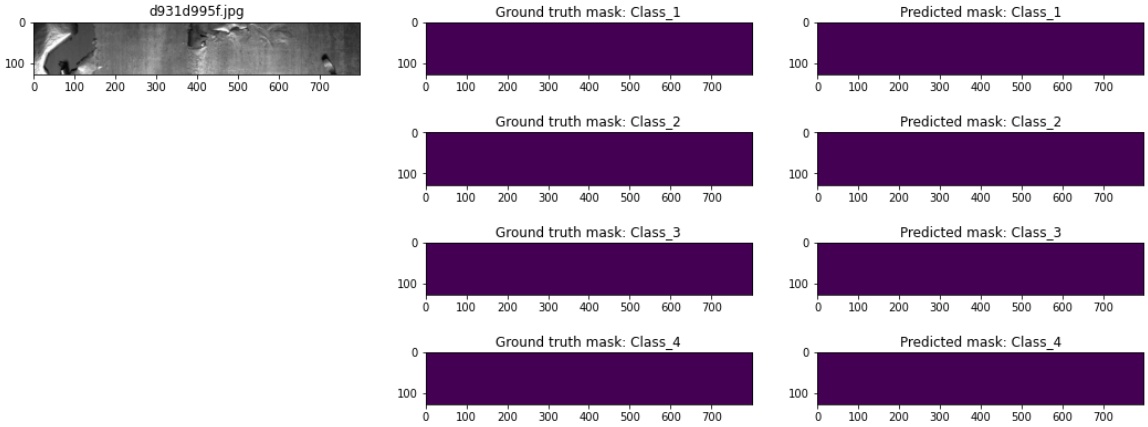
```
visualize_defects(train_data,model_best)
```


f480125b7.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


c0d23ec4a.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

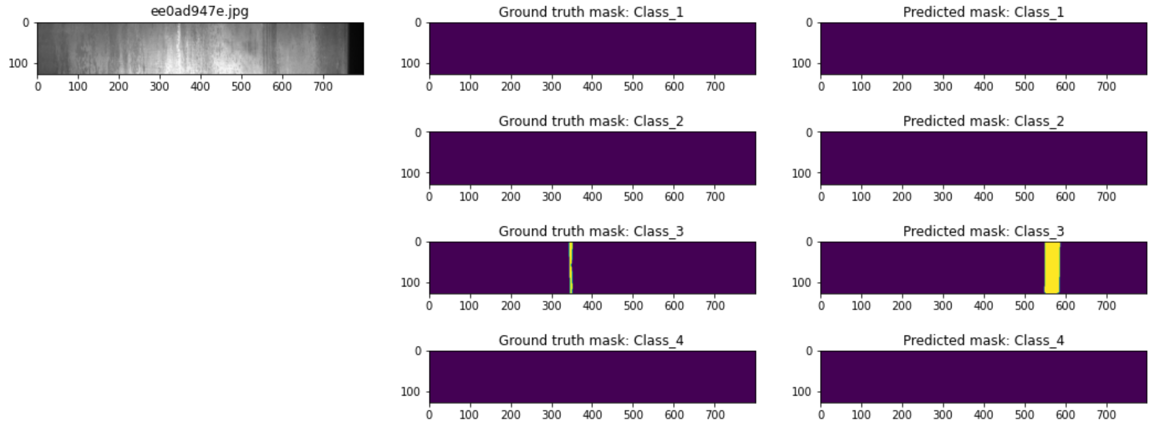

60f9f7ff2.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

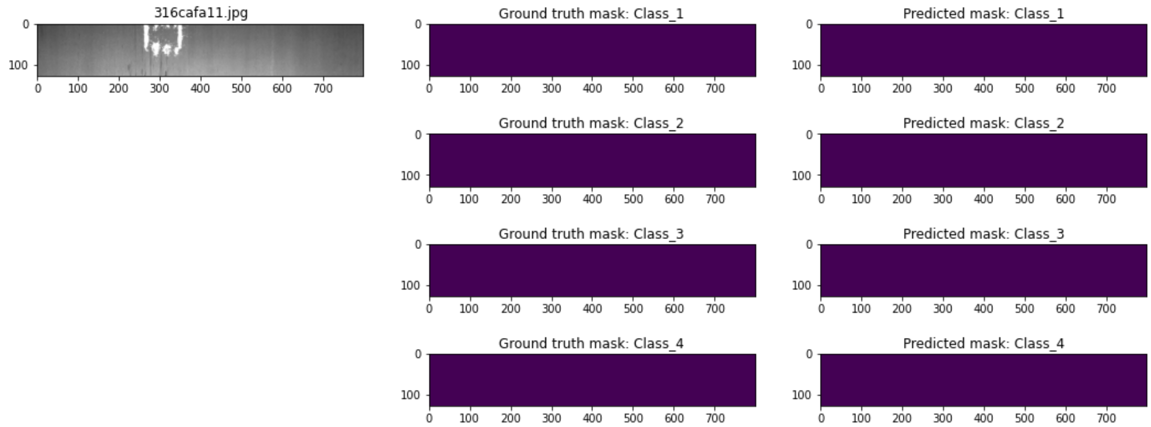

711d59ca6.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

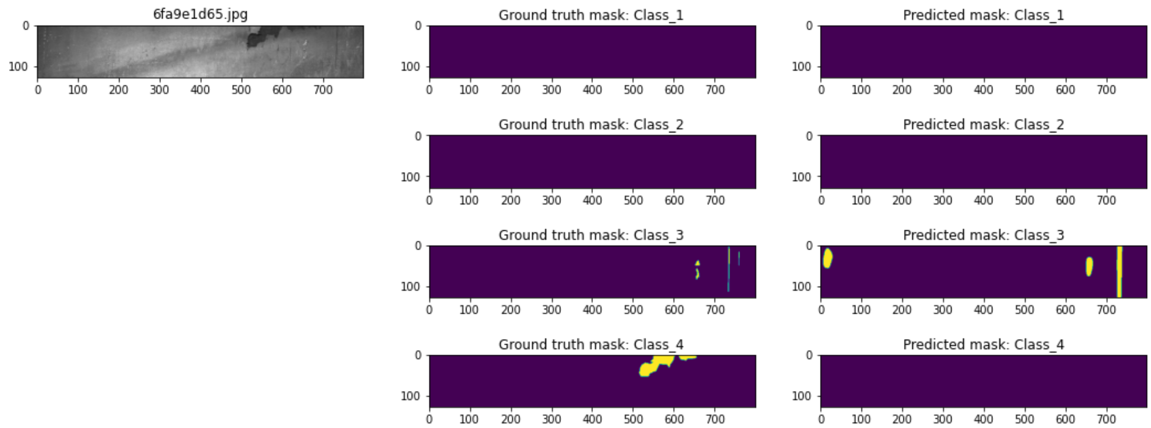

d931d995f.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

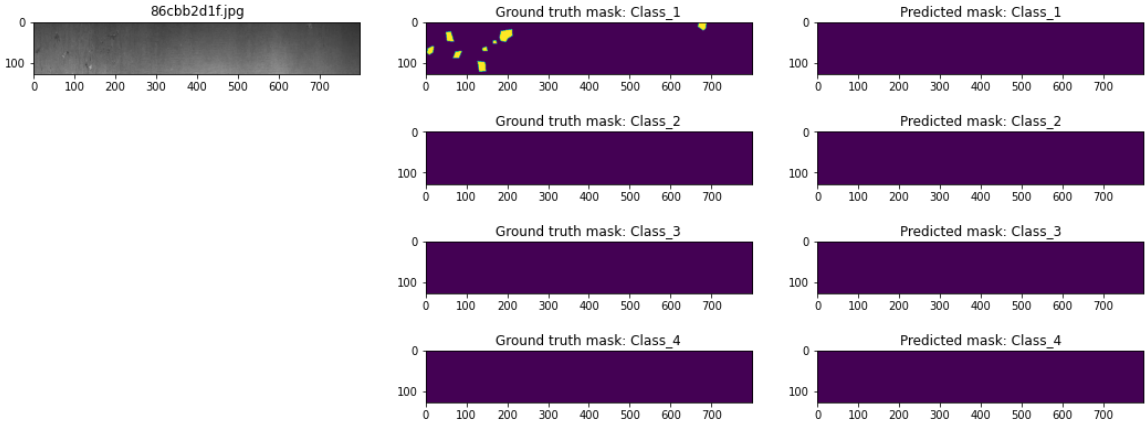
In [0]:

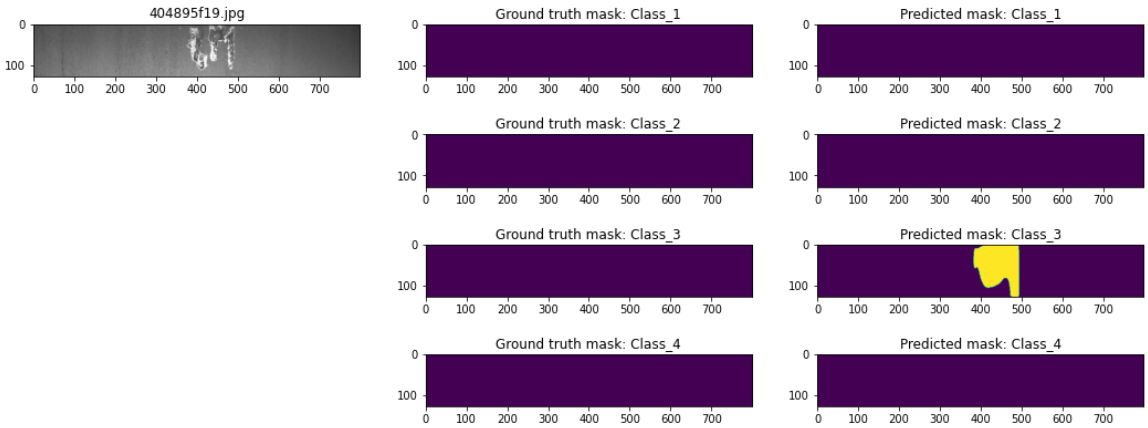
```
visualize_defects(train_data,model_best)
```


ee0ad947e.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


316cafa11.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


6fa9e1d65.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

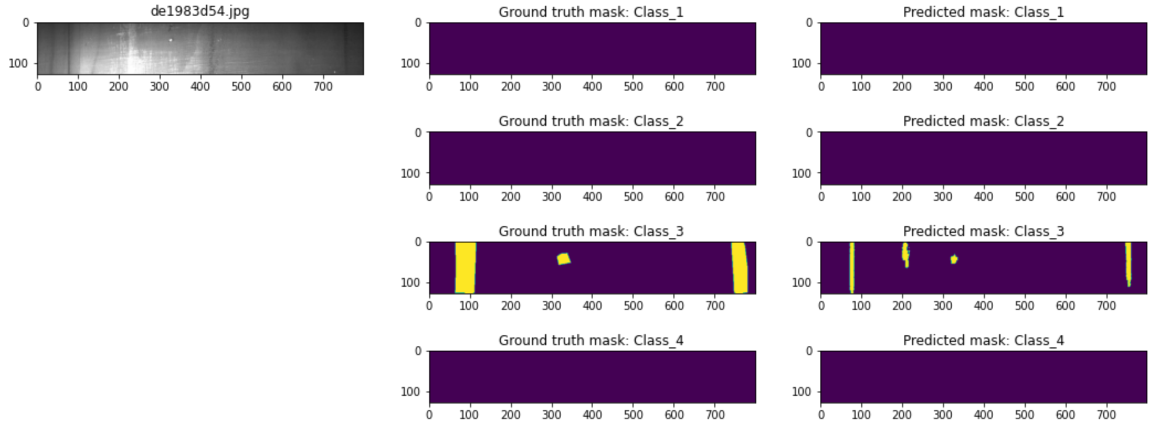




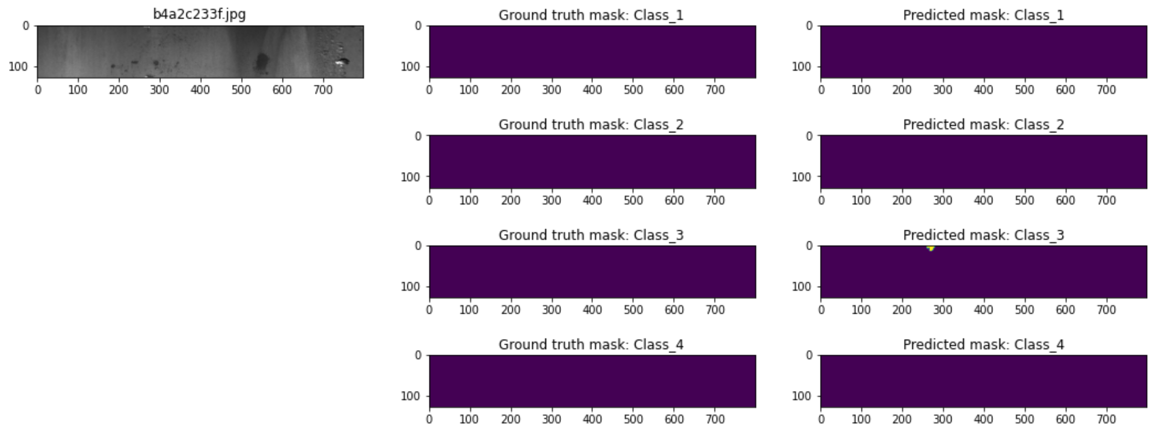
Validation set

In [0]:

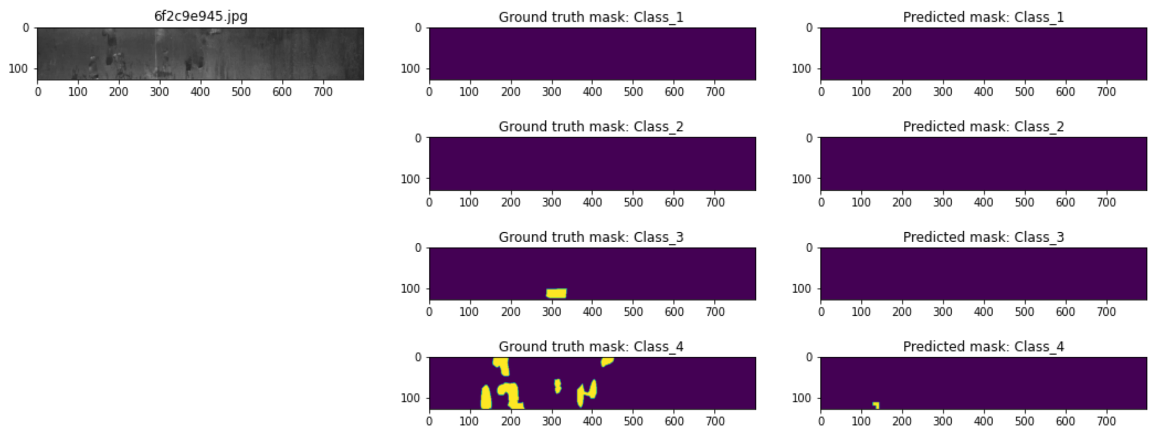
```
visualize_defects(cv_data,model_best)
```


de1983d54.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

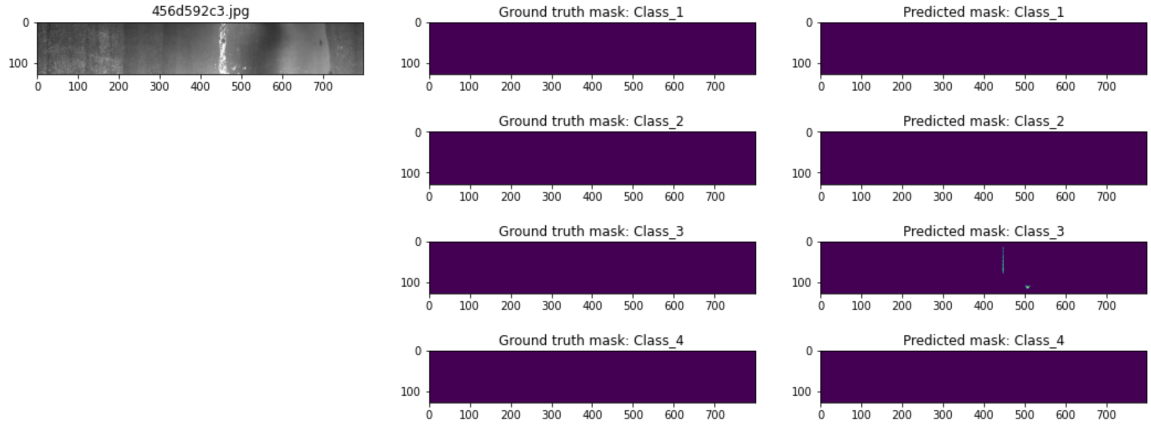
--

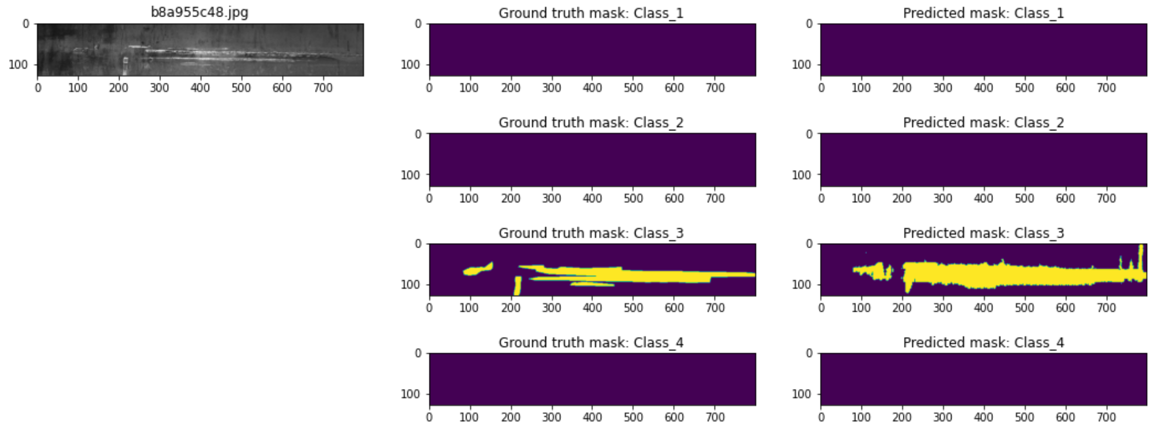

b4a2c233f.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

--


6f2c9e945.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

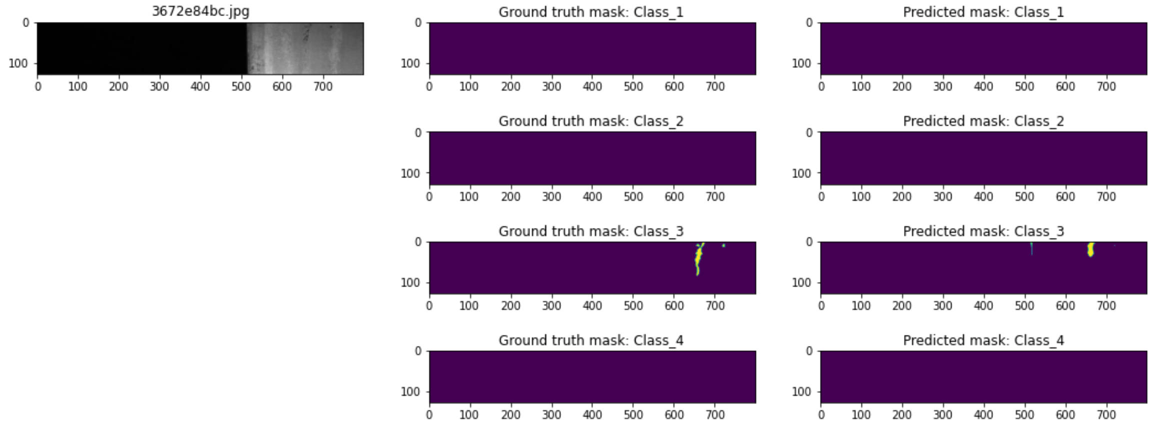
--

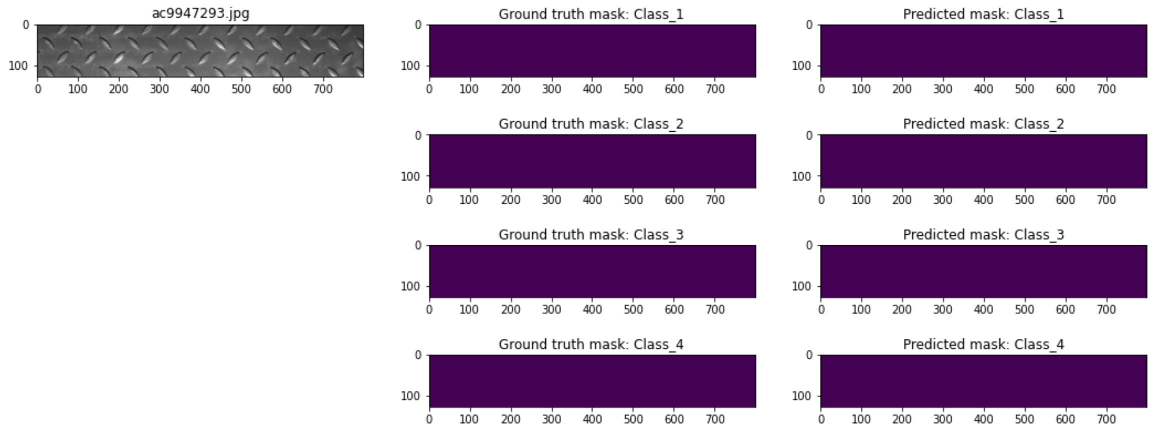

456d592c3.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

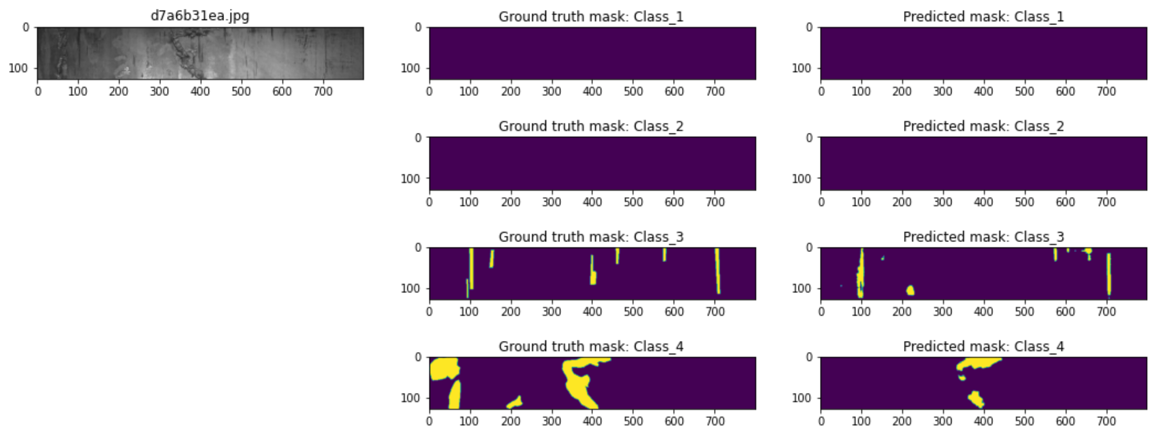

b8a955c48.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4

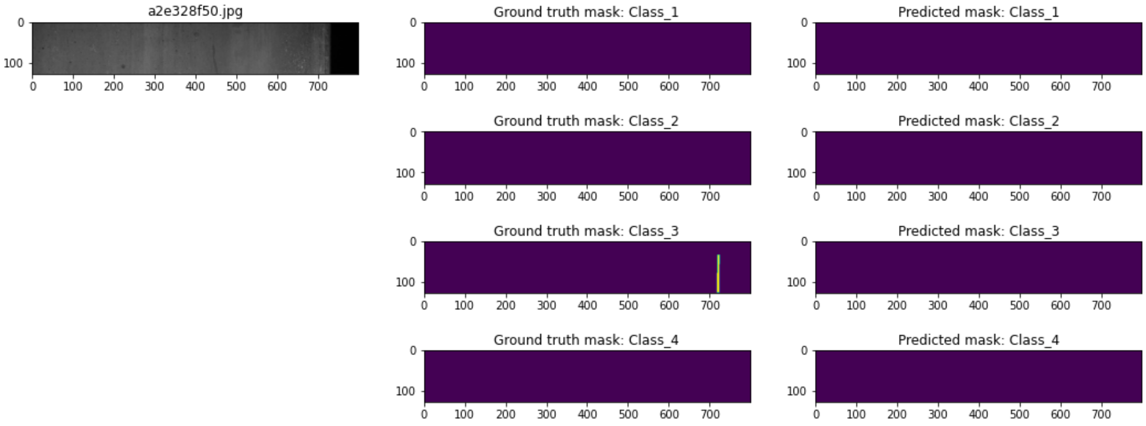
In [0]:

```
visualize_defects(cv_data,model_best)
```

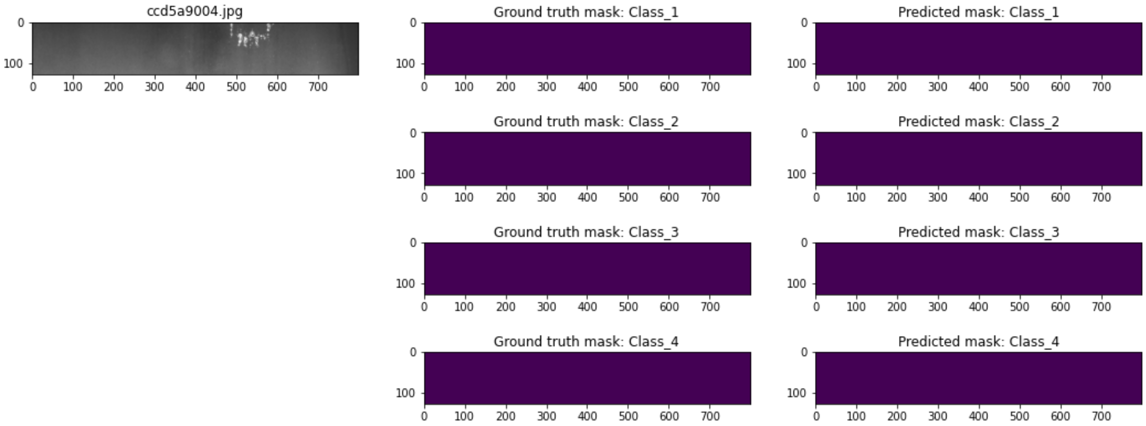

3672e84bc.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


ac9947293.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4


d7a6b31ea.jpg
Ground truth mask: Class_1
Predicted mask: Class_1
Ground truth mask: Class_2
Predicted mask: Class_2
Ground truth mask: Class_3
Predicted mask: Class_3
Ground truth mask: Class_4
Predicted mask: Class_4




```
------------------------------------------------------------------------------------------
```




```
------------------------------------------------------------------------------------------
```

3.6 Predicting defects on raw test images

Redefining the FCN8 architecture for original image size of 256x1600

In [0]:

```
#https://keras.io/models/model/#fit_generator
model= fcn_8(img_shape= (256,1600,3), base= 4, dropout= 0.3, n_classes= 4)
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
model.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
input_5 (InputLayer)         (None, 256, 1600, 3)      0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 1600, 16      448
input_5[0][0]
_________________________________________________________________
batch_normalization_61 (BatchNo (None, 256, 1600, 16      64
block1_conv1[0][0]
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 1600, 16      2320
batch_normalization_61[0][0]
_________________________________________________________________
batch_normalization_62 (BatchNo (None, 256, 1600, 16      64
block1_conv2[0][0]
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 800, 16)      0
batch_normalization_62[0][0]
_________________________________________________________________
dropout_29 (Dropout)         (None, 128, 800, 16)      0
block1_pool[0][0]
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 800, 32)      4640
dropout_29[0][0]
_________________________________________________________________
batch_normalization_63 (BatchNo (None, 128, 800, 32)      128
block2_conv1[0][0]
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 800, 32)      9248
batch_normalization_63[0][0]
_________________________________________________________________
batch_normalization_64 (BatchNo (None, 128, 800, 32)      128
block2_conv2[0][0]
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 400, 32)       0
batch_normalization_64[0][0]
_________________________________________________________________
dropout_30 (Dropout)         (None, 64, 400, 32)       0
block2_pool[0][0]
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 400, 64)       18496
dropout_30[0][0]
_________________________________________________________________
```

```
_____________________________
batch_normalization_65 (BatchNo (None, 64, 400, 64)  256         block3_co
nv1[0][0]
_____________________________________________________________________________
_____________________________
block3_conv2 (Conv2D)           (None, 64, 400, 64)  36928       batch_nor
malization_65[0][0]
_____________________________________________________________________________
_____________________________
batch_normalization_66 (BatchNo (None, 64, 400, 64)  256         block3_co
nv2[0][0]
_____________________________________________________________________________
_____________________________
block3_conv3 (Conv2D)           (None, 64, 400, 64)  36928       batch_nor
malization_66[0][0]
_____________________________________________________________________________
_____________________________
batch_normalization_67 (BatchNo (None, 64, 400, 64)  256         block3_co
nv3[0][0]
_____________________________________________________________________________
_____________________________
block3_pool (MaxPooling2D)      (None, 32, 200, 64)  0           batch_nor
malization_67[0][0]
_____________________________________________________________________________
_____________________________
dropout_31 (Dropout)            (None, 32, 200, 64)  0           block3_po
ol[0][0]
_____________________________________________________________________________
_____________________________
block4_conv1 (Conv2D)           (None, 32, 200, 128) 73856       dropout_3
1[0][0]
_____________________________________________________________________________
_____________________________
batch_normalization_68 (BatchNo (None, 32, 200, 128) 512         block4_co
nv1[0][0]
_____________________________________________________________________________
_____________________________
block4_conv2 (Conv2D)           (None, 32, 200, 128) 147584      batch_nor
malization_68[0][0]
_____________________________________________________________________________
_____________________________
batch_normalization_69 (BatchNo (None, 32, 200, 128) 512         block4_co
nv2[0][0]
_____________________________________________________________________________
_____________________________
block4_conv3 (Conv2D)           (None, 32, 200, 128) 147584      batch_nor
malization_69[0][0]
_____________________________________________________________________________
_____________________________
batch_normalization_70 (BatchNo (None, 32, 200, 128) 512         block4_co
nv3[0][0]
_____________________________________________________________________________
_____________________________
block4_pool (MaxPooling2D)      (None, 16, 100, 128) 0           batch_nor
malization_70[0][0]
_____________________________________________________________________________
_____________________________
dropout_32 (Dropout)            (None, 16, 100, 128) 0           block4_po
ol[0][0]
_____________________________________________________________________________
_____________________________
```

```
block5_conv1 (Conv2D)           (None, 16, 100, 128) 147584      dropout_3
2[0][0]
__________________________________________________________________________
_____________________
batch_normalization_71 (BatchNo (None, 16, 100, 128) 512         block5_co
nv1[0][0]
__________________________________________________________________________
_____________________
block5_conv2 (Conv2D)           (None, 16, 100, 128) 147584      batch_nor
malization_71[0][0]
__________________________________________________________________________
_____________________
batch_normalization_72 (BatchNo (None, 16, 100, 128) 512         block5_co
nv2[0][0]
__________________________________________________________________________
_____________________
block5_conv3 (Conv2D)           (None, 16, 100, 128) 147584      batch_nor
malization_72[0][0]
__________________________________________________________________________
_____________________
batch_normalization_73 (BatchNo (None, 16, 100, 128) 512         block5_co
nv3[0][0]
__________________________________________________________________________
_____________________
block5_pool (MaxPooling2D)      (None, 8, 50, 128)   0           batch_nor
malization_73[0][0]
__________________________________________________________________________
_____________________
dropout_33 (Dropout)            (None, 8, 50, 128)   0           block5_po
ol[0][0]
__________________________________________________________________________
_____________________
conv6 (Conv2D)                  (None, 8, 50, 1024)  6423552     dropout_3
3[0][0]
__________________________________________________________________________
_____________________
batch_normalization_74 (BatchNo (None, 8, 50, 1024)  4096        conv6[0]
[0]
__________________________________________________________________________
_____________________
dropout_34 (Dropout)            (None, 8, 50, 1024)  0           batch_nor
malization_74[0][0]
__________________________________________________________________________
_____________________
conv7 (Conv2D)                  (None, 8, 50, 1024)  1049600     dropout_3
4[0][0]
__________________________________________________________________________
_____________________
batch_normalization_75 (BatchNo (None, 8, 50, 1024)  4096        conv7[0]
[0]
__________________________________________________________________________
_____________________
dropout_35 (Dropout)            (None, 8, 50, 1024)  0           batch_nor
malization_75[0][0]
__________________________________________________________________________
_____________________
conv2d_9 (Conv2D)               (None, 16, 100, 4)   516         dropout_3
2[0][0]
__________________________________________________________________________
_____________________
conv2d_transpose_13 (Conv2DTran (None, 16, 100, 4)   16388       dropout_3
```

```
5[0][0]
__________________________________________________________________________________
_____________________
add_9 (Add)                     (None, 16, 100, 4)   0           conv2d_9
[0][0]
                                                                 conv2d_tr
anspose_13[0][0]
__________________________________________________________________________________
_____________________
conv2d_10 (Conv2D)              (None, 32, 200, 4)   260         dropout_3
1[0][0]
__________________________________________________________________________________
_____________________
conv2d_transpose_14 (Conv2DTran (None, 32, 200, 4)   68          add_9[0]
[0]
__________________________________________________________________________________
_____________________
add_10 (Add)                    (None, 32, 200, 4)   0           conv2d_10
[0][0]
                                                                 conv2d_tr
anspose_14[0][0]
__________________________________________________________________________________
_____________________
conv2d_transpose_15 (Conv2DTran (None, 256, 1600, 4) 1028        add_10[0]
[0]
==================================================================================
========================
Total params: 8,424,612
Trainable params: 8,418,404
Non-trainable params: 6,208
__________________________________________________________________________________
_____________________
```

In [0]:

```
model.set_weights(model_best.get_weights())
```

Predicting on full 256x1600 raw test images

In [0]:

```
# Predicting on test images
from tqdm import tqdm
import cv2
data_path = '/content/' + 'test_images/'
files = list(os.listdir(data_path))
rle_lst = [] #list to store defect in run length encoding format
img_classId= [] #list to store Image ID + classId

for f in tqdm(files):
    X = np.empty((1,256,1600,3),dtype=np.float32)
    img = cv2.imread(data_path + f)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    X[0,] = img
    mask = model.predict(X)
    rle_m = np.empty((256,1600),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))
```

100%|██████████| 5506/5506 [05:48<00:00, 15.82it/s]

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}
import pandas as pd
output_df = pd.DataFrame(output)
output_df.to_csv('submission_fcn8_256x1600.csv', index=False)
```

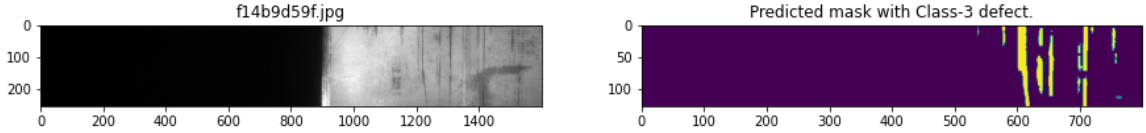
With this submission, I got a private dice coefficient score of 0.82818 & a public score of 0.84935.

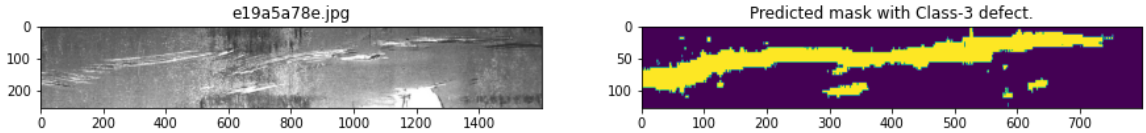
3.7 Visualizing defects of raw test images(256x1600)

Please refer to the "visualize_defects_test" function in Utility functions section.

In [0]:

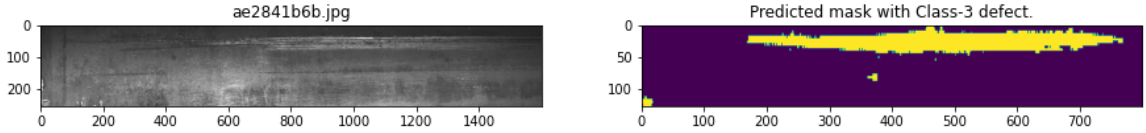
```
visualize_defects_test(output_df,5)
```

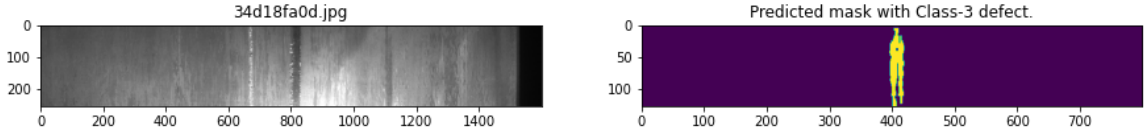


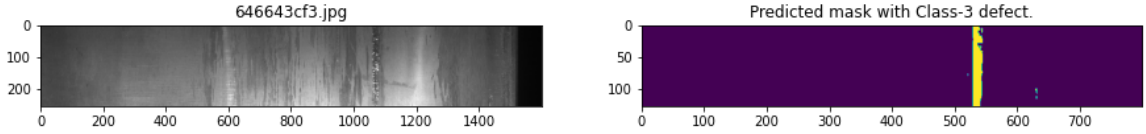


In [0]:

```
visualize_defects_test(output_df,5)
```

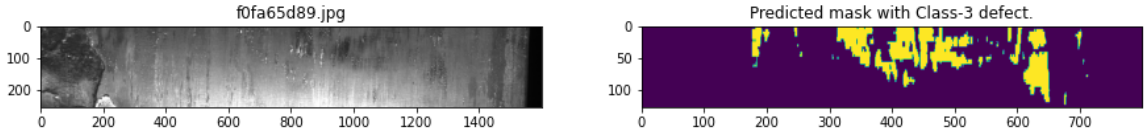






In [0]:

```
visualize_defects_test(output_df,5)
```



Results Summary

In [2]:

```
#Ref: http://zetcode.com/python/prettytable/
from prettytable import PrettyTable


x = PrettyTable()
x.field_names = ["Architecture","Test Dice_coeff-Private","Test Dice_coeff-Public"]
x.add_row(["UNET",0.81171,0.80041])
x.add_row(["SEGNET",0.83437,0.84741])
x.add_row(["FCN-8",0.82818,0.84935])
print(x)
```

```
+--------------+-------------------------+-----------------------+
| Architecture | Test Dice_coeff-Private | Test Dice_coeff-Public |
+--------------+-------------------------+-----------------------+
|     UNET     |         0.81171         |        0.80041         |
|    SEGNET    |         0.83437         |        0.84741         |
|    FCN-8     |         0.82818         |        0.84935         |
+--------------+-------------------------+-----------------------+
```

Conclusion

- Countered the issue of class imbalance by assigning class weights (calculated using SKLearn) during the training stage
- Model training for all architectures was done on resized 128x800 images considering efficiency and computational power
- Model prediction on full sized raw test images (256x1600) gave better scores compared to prediction on half images (128x800)
- Highest score of 0.83437 on private leaderboard & 0.84935 on public leaderboard was obtained with SEGNET & FCN-8 architecture respectively. These scores were better compared to the basic U-NET architecture which was the first cut aproach.

Steps followed to solve this case study(in brief):

1. The first basic step was to carry out EDA on the defect data. This included analysing images in train & test folder to get a gist of defective & non-defective images, knowing the basic properties of the given images, analysing response label, determining class weights, checking number of labels tagged to each image & observing images with defects belonging to each class.
2. EDA was then followed by Data preparation stage where a master csv file was prepared in which each row represented a image with all possible defects that image possessed.
3. The final data was split into train & CV data in the ratio 85:15 and basic utility functions were defined.
4. Next was the data generator phase where train & validation data was generated in order to feed the keras models with augmented images (in case of train) and ground truth masks.
5. Once the flow of input was set, it was time to implement various segmentation architectures, This included UNET, SEGNET & FCN-8.
6. For each architecture, the performance was checked on the validation image data and few predicted masks were observed.
7. Predictions were made on full sized (i.e 256x1600) raw test images and the predicted masks were observed.
8. Predictions on test images were uploaded on kaggle to get the dice coefficient score