Severstal: Steel Defect Detection

Steel is one of the most important building materials of modern times. The physical properties of steel include high strength, low weight, durability, ductility and resistance to corrosion. Due to these properties of steel, buildings are resistant to natural and man-made wear which has made the material ubiquitous around the world.

Severstal is among the top 50 producers of steel in the world and produced 12.04 and 11.8 Million tonnes of steel in 2018 and 2019 respectively. It is one among Russia's biggest players in efficient steel mining and production. The company recently created a hybrid Data Lake as part of its digital strategy to secure the Company's competitive advantages in the long-term. The infrastructure is designed to store Company functional data files for subsequent processing and use in Severstal's data analysis, machine learning and artificial intelligence projects. Severstal is now looking to machine learning to improve automation, increase efficiency, and maintain high quality in their production.

1. Business Problem

One of the key products of Severstal is steel sheets. The production process of flat sheet steel is delicate. From heating and rolling, to drying and cutting, several machines touch flat steel by the time it's ready to ship. To ensure quality in the production of steel sheets, today, Severstal uses images from high frequency cameras to power a defect detection algorithm.

Through this competition, Severstal expects the AI community to improve the algorithm by **localizing and classifying surface defects on a steel sheet**.

1.1. Business objectives and constraints

1. A defective sheet most be predicted as defective, since there would be serious concerns about quality if we misclassify a defective sheet as non-defective. i.e. high recall value for each of the classes is needed.
2. No strict latency concerns.

1.2. Sources / References

Kaggle competition page : https://www.kaggle.com/c/severstal-steel-defect-detection/overview

References :

- https://arxiv.org/pdf/1505.04597.pdf
- https://www.kaggle.com/cdeotte/keras-unet-with-eda
- https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient
- https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

2. Machine Learning Probelm

2.1. Mapping the business problem to an ML problem

Our task is to

1. Detect/localize the defects in a steel sheet using image segmentation and
2. Classify the detected defects into one or more classes from [1, 2, 3, 4]

Therefore, it is a combination of image segementation and multiclass classification.

2.2. Performance metric

Evaluation metric used is the mean Dice coefficient. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. The formula is given by:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

where X is the predicted set of pixels and Y is the ground truth.

[Read more about Dice Coefficient]

2.3. Data Overview

We have been given a zip folder of size 2GB which contains the following:

- train_images/ - folder containilning 12,568 training images (.jpg files)
- test_images/ - folder containing 5506 test images (.jpg files). We need to detect and localize defect in these images.
- train.csv - training annotations which provide segments for defects belonging to 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

Refer to section 3: EDA for more details about data.

In [1]:

```
% cd /content/sample_data/Data
```

```
/content/sample_data/Data
```

Downloading and extracting data

In [0]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36" --header="A
ccept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9,hi;q=0.8" --header="Referer: http
s://www.kaggle.com/" "https://storage.googleapis.com/kaggle-competitions-data/kaggle-
v2/14241/862020/bundle/archive.zip?GoogleAccessId=web-data@kaggle-
161607.iam.gserviceaccount.com&Expires=1588779165&Signature=p8ocGw57q6rPTefACaYTOvrfIjZpdsHhwO6v5S
nNOLtUiXBr69RIbU4vYI6PcDL23eIi1UfFHDFy05dPyqXFrBuyVKYVfNJYhkb7yN0Aike7RYkk03IECp5XoIW9nb1R%2FBCEon
eLSz5oBvbjox4GlX%2FqczPtRZ9cRxKcNOXhWFJFWBW1o5rCULR2lpyuTvcoTb6jU1bR8OUZ0B1lnP22LDEv2qAQPScF7J828f
J03kW5B%2FDcZ%2FL%2FhWScH7AKvGTOOuBxCrNo5I%2FDbWR6VrRbOgMwBT4k97yzOo%2FQd1oLzl4GGgorcPqeWLmCLe4nSZ
dmMQ%3D%3D&response-content-disposition=attachment%3B+filename%3Dseverstal-steel-defect-
detection.zip" -c -O 'severstal-steel-defect-detection.zip'
```

In [3]:

```
from zipfile import ZipFile
file_name="severstal-steel-defect-detection.zip"

with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

```
Done
```

Importing libraries

In [0]:

```
## Importing required packages
import warnings
warnings.filterwarnings("ignore")
from datetime import datetime
import os
import gc
import pickle

from tqdm import tqdm import tqdm_notebook as tqdm
```

```
import pandas as pd
import numpy as np
import math
from numpy import asarray
import cv2
from os import listdir
import random

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

from PIL import Image

from sklearn import metrics

from collections import Counter
from collections import defaultdict
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import binary_crossentropy

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint,
LearningRateScheduler,Callback
```

In [0]:

```
tf.keras.backend.clear_session()
```

3. Loading data

In [5]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%
b&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
www.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly
ttps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

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

In [6]:

```
# Loading pre-processed data
data = pd.read_csv("/content/drive/My Drive/CaseStudy2/prep_data.csv")
data.head()
```

Out[6]:

	ImageId	Defect_1	Defect_2	Defect_3	Defect_4
0	0002cc93b.jpg	29102 12 29346 24 29602 24 29858 24 30114 24 3...	NaN	NaN	NaN
1	00031f466.jpg	NaN	NaN	NaN	NaN
2	000418bfc.jpg	NaN	NaN	NaN	NaN
3	000789191.jpg	NaN	NaN	NaN	NaN
4	0007a71bf.jpg	NaN	NaN	18661 28 18863 82 19091 110 19347 110 19603 11...	NaN

In [7]:

```
# Replace NAs with blank spaces
data.fillna('', inplace=True)
data.head()
```

Out[7]:

	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...	

3.1. Train, CV split 85:15

In [8]:

```
#splitting the data into train & cv
from sklearn.model_selection import train_test_split
train_data, cv_data = train_test_split(data, test_size=0.15, random_state=42)
print(train_data.shape)
print(cv_data.shape)
```

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

3.2. Functions for converting RLE encoded pixels to masks and vice-versa

In [0]:

```
# https://www.kaggle.com/titericz/building-and-visualizing-masks

def rle2mask(rle):
    # CONVERT RLE TO MASK
    if (pd.isnull(rle))|(rle==''):
        return np.zeros((128,800) ,dtype=np.uint8)

    height= 256
    width = 1600
    mask= np.zeros( width*height ,dtype=np.uint8)

    array = np.asarray([int(x) for x in rle.split()])
    starts = array[0::2]-1
    lengths = array[1::2]
    for index, start in enumerate(starts):
        mask[int(start):int(start+lengths[index])] = 1

    return mask.reshape( (height,width), order='F' )[::2,::2]


# to convert masks to run length encoded values
def mask2rle(img):
    '''
    img: numpy array containing ones and zeros as pixel values, 1 - mask, 0 - background
    Returns String run length ecoded pixels
    '''
    pixels= img.T.flatten() # Convert nd-array to 1d-array (numbering of pixels is from top to
bottom)
    pixels = np.concatenate([[0], pixels, [0]]) # Adding zeros at the start and end so that if ther
e's mask at the first/last pixel, it gets detected.
```

```
    runs = np.where(pixels[1:] != pixels[:-1])[0] + 1 # Detect all changing pixels (where pixel values changes, either 0 -> 1 or 1 -> 0)

    # To get RLE, we need start pixels and run lengths
    # Start pixels are the pixels where change 0 -> 1 occurs, i.e. pixels at even indices
    # Run length is the pixel distance between two consecutive changing pixels. So, run lengths = odd indices - even indices
    runs[1::2] -= runs[::2]

    return ' '.join(str(x) for x in runs)
```

3.3. Generating masks and saving to drive

In [0]:

```
indices = data.index
y = np.empty((data.shape[0], 128, 800, 4), dtype=np.uint8)
for i, f in enumerate(data['ImageId']):
    f = f.split('.')[0]
    #run-length encoding on the pixel values
    for j in range(4):
        y[i,:,:,j] = rle2mask(data['Defect_'+str(j+1)].iloc[indices[i]])
        cv2.imwrite(f'train_masks/{f}_mask{j+1}.png', y[i,:,:,j])
```

In [11]:

```
# Check if masks are generated
len(os.listdir('/content/sample_data/Data/train_masks'))
```

Out[11]:

```
50272
```

3.4. Generating data for TF model

3.4.1. Creating path lists

In [0]:

```
train_ids = train_data['ImageId'].values
train_image_paths = ['train_images/'+i for i in train_ids]

train_label_paths = [['train_masks/'+i.split('.')[0]+'_mask1.png',
                     'train_masks/'+i.split('.')[0]+'_mask2.png',
                     'train_masks/'+i.split('.')[0]+'_mask3.png',
                     'train_masks/'+i.split('.')[0]+'_mask4.png'] for i in train_ids]
```

In [0]:

```
val_ids = cv_data['ImageId'].values
val_image_paths = ['train_images/'+i for i in val_ids]

val_label_paths = [['train_masks/'+i.split('.')[0]+'_mask1.png',
                     'train_masks/'+i.split('.')[0]+'_mask2.png',
                     'train_masks/'+i.split('.')[0]+'_mask3.png',
                     'train_masks/'+i.split('.')[0]+'_mask4.png'] for i in val_ids]
```

In [14]:

```
print(len(train_image_paths), len(train_label_paths), len(train_label_paths[0]))
```

```
10682 10682 4
```

3.4.2. Data generator using tf.data

In [0]:

```
tf.random.set_seed(42)
def tfdata_generator(images, labels, is_training, batch_size=16):
    '''Construct a data generator using tf.Dataset'''

    def parse_function(filename, labels):

        #reading image
        image_string = tf.io.read_file(filename) # read as string of pixel values
        image = tf.image.decode_jpeg(image_string, channels=3) # decode image as tensor of dtype ui
nt8

        image = tf.image.convert_image_dtype(image, tf.float32) # convert to float values in range
[0, 1]
        image = tf.image.resize(image, [128, 800]) #resize to desired size

        #reading label masks
        y = tf.zeros((128,800,1), dtype=tf.uint8)
        for j in range(4):
            mask_string = tf.io.read_file(labels[j])
            mask = tf.image.decode_jpeg(mask_string)
            mask = tf.image.convert_image_dtype(mask, tf.uint8)

            y = tf.concat([y, mask], 2)


        return image, y[:,:,1:]

    def flip(image,labels):

        image = tf.image.random_flip_left_right(image, seed=1)
        labels = tf.image.random_flip_left_right(labels, seed=1)
        image = tf.image.random_flip_up_down(image, seed=1)
        labels = tf.image.random_flip_up_down(labels, seed=1)

        return image, labels

    def color(image,labels):
        image = tf.image.random_hue(image, 0.05)
        image = tf.image.random_saturation(image, 0.4, 1.2)
        image = tf.image.random_brightness(image, 0.05)
        image = tf.image.random_contrast(image, 0.4, 1.2)
        return image, labels

    dataset = tf.data.Dataset.from_tensor_slices((images, labels))

    if is_training:
        dataset = dataset.shuffle(5000)  # depends on sample size

    # Transform and batch data at the same time
    dataset = dataset.map(parse_function, num_parallel_calls=4)

    augmentations = [flip, color]

    if is_training:
      for f in augmentations:
        if tf.random.uniform([1], 0, 1)>0.6:
            dataset = dataset.map(f, num_parallel_calls=4)

    # dataset = dataset.repeat()

    dataset = dataset.batch(batch_size).prefetch(tf.data.experimental.AUTOTUNE)
    return dataset
```

In [0]:

```
tf_image_generator = tfdata_generator(train_image_paths, train_label_paths, is_training=True,
batch_size=8)
```

3.5. Defining metric and loss function

In [0]:

```
from tensorflow.keras import backend as K
from tensorflow.keras.losses import binary_crossentropy

def dice_coef(y_true, y_pred, smooth=1):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

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

def  dice_loss(y_true, y_predict):
    return (1-dice_coef(y_true, y_predict))
```

4. Model

This model is based on the paper [Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation]() by Google. The model is called DeepLabv3+.

4.1. Defining building blocks using subclass Layer and Model

Writing custom layers and models with Keras

https://www.tensorflow.org/tutorials/customization/custom_layers

https://www.tensorflow.org/guide/keras/custom_layers_and_models#the_model_class

In [0]:

```
import tensorflow as tf
from tensorflow.keras import backend as K

# from keras.applications import imagenet_utils
from tensorflow.keras.utils import get_file, plot_model
from tensorflow.keras import layers, Model
import tensorflow.keras as keras
```

In [0]:

```
class Conv2D_custom(layers.Layer):

    ''' args:

            filters: Integer, the number of filters in convolution
            prefix: String, name of the block of which this layer is a part
            stride: Integer, stride for convolution, default=1
            kernel_size: Integer, size of kernel for convolution, default=3
            rate: Integer, atrous rate for convolution, default=1

        Input: 4D tensor with shape (batch, rows, cols, channels)
        Output: 4D tensor with shape (batch, new_rows, new_cols, filters)
    '''

    def __init__(self, filters, prefix='', stride=1, kernel_size=3, rate=1):
        super(Conv2D_custom, self).__init__()

        self.stride = stride
        # manual padding when stride!=1
        if stride!=1:
            #effective kernel size = kernel_size + (kernel_size - 1) * (rate - 1)
            n_pads = (kernel_size + (kernel_size - 1) * (rate - 1) - 1) // 2
            self.zeropad = layers.ZeroPadding2D(padding=n_pads)

        self.conv_2d = layers.Conv2D(filters, kernel_size=kernel_size, strides= stride,
dilation_rate=rate,
                                padding='same' if stride==1 else 'valid', name=prefix + 'Conv2D_custom'

    def call(self, x):
```

```
        if self.stride != 1:
            x = self.zeropad(x)

        x = self.conv_2d(x)
        return x
```

In [0]:

```
class SeparableConv_BN(Model):

    ''' Separable convolutions consist in first performing a depthwise spatial convolution
        (which acts on each input channel separately) followed by a pointwise convolution which mi
xes together the resulting output channels.

        - This is a custom implementation of SeparableConv2D layer.
        - To be used in encoder (a modified Xception block) of DeeplabV3+.

        The difference between this implementation and tf.keras.layers.SeparableConv2D is that
        here, extra batch normalization and ReLU are added after each 3×3 depthwise convolution

        args:

            filters: Integer, the number of output filters in pointwise convolution
            prefix: String, name of the block of which this layer is a part
            stride: Integer, stride for depthwise convolution, default=1
            kernel_size: Integer, size of kernel for depthwise convolution, default=3
            rate: Integer, atrous rate for depthwise convolution, default=1
            depth_activation: Bool, flag to use activation between depthwise & pointwise convolutio
ns

        Input: 4D tensor with shape (batch, rows, cols, channels)
        Output: 4D tensor with shape (batch, new_rows, new_cols, filters)
    '''

    def __init__(self, filters, prefix='', stride=1, kernel_size=3, rate=1, depth_activation=False)
:
        super(SeparableConv_BN, self).__init__()

        self.stride = stride
        self.depth_activation = depth_activation
        # manual padding size when stride!=1
        if stride!=1:
            #effective kernel size = kernel_size + (kernel_size - 1) * (rate - 1)
            n_pads = (kernel_size + (kernel_size - 1) * (rate - 1) - 1) // 2
            self.zeropad = layers.ZeroPadding2D(padding=n_pads)

        self.depthwise_conv = layers.DepthwiseConv2D(kernel_size=kernel_size, strides= stride, dila
tion_rate=rate,
                                padding='same' if stride==1 else 'valid', name=prefix + '_depthW')
        self.batchnorm_d = layers.BatchNormalization(name=prefix + '_depthW_BN')

        self.pointwise_conv = layers.Conv2D(filters, kernel_size=1, padding='same', name=prefix + '
_pointW')
        self.batchnorm_p = layers.BatchNormalization(name=prefix + '_pointW_BN')

    def call(self, x):
        if self.stride != 1:
            x = self.zeropad(x)

        if not self.depth_activation:
            x = tf.nn.relu(x)

        x = self.depthwise_conv(x)
        x = self.batchnorm_d(x)
        if self.depth_activation:
            x = tf.nn.relu(x)
        x = self.pointwise_conv(x)
        x = self.batchnorm_p(x)
        if self.depth_activation:
            x = tf.nn.relu(x)

        return x
```

In [0]:

```
class Xception_Block(Model):
    ''' Basic building block of DeepLabV3+ encoder (modified Xception) network.
        It consists of 3 SeparableConv_BN layers.

        args:
            depth_list: list of 3 Integers, number of filters in each SeparableConv_BN.
            prefix: String, prefix before name of the layer
            short_path_type: String, one of {'conv','sum'} default=None; type of shortcut
connection between input and output of the block
            stride: Integer, stride for depthwise convolution in last(3rd) layer
            rate: Integer, atrous rate for depthwise convolution
            depth_activation: Bool, flag to use activation between depthwise & pointwise convoluti
ns
            return_skip: Bool, flag to return additional tensor after 2 SepConvs for decoder
    '''
    def __init__(self, depth_list, prefix='', residual_type=None, stride=1, rate=1,
depth_activation=False, return_skip=False):
        super(Xception_Block, self).__init__()

        self.sepConv1 = SeparableConv_BN(filters=depth_list[0], prefix=prefix +'_sepConv1', stride=
1, rate=rate, depth_activation=depth_activation)
        self.sepConv2 = SeparableConv_BN(filters=depth_list[1], prefix=prefix +'_sepConv2', stride=
1, rate=rate, depth_activation=depth_activation)
        self.sepConv3 = SeparableConv_BN(filters=depth_list[2], prefix=prefix +'_sepConv3', stride=
stride, rate=rate, depth_activation=depth_activation)

        if residual_type == 'conv':
            self.conv2D = Conv2D_custom(depth_list[2], prefix=prefix+'_conv_residual',
stride=stride, kernel_size=1, rate=1)
            self.batchnorm_res = layers.BatchNormalization(name=prefix + '_BN_residual')

        self.return_skip = return_skip
        self.residual_type = residual_type

    def call(self, x):
        output = self.sepConv1(x)
        output = self.sepConv2(output)
        skip = output # skip connection to decoder
        output= self.sepConv3(output)

        if self.residual_type == 'conv':
            res = self.conv2D(x)
            res = self.batchnorm_res(res)
            output += res
        elif self.residual_type == 'sum':
            output +=  x
        else:
            if(self.residual_type):
                raise ValueError('Arg residual_type should be one of {conv, sum}')

        if self.return_skip:
            return output, skip

        return output
```

4.2. DeepLabV3+ architecture

In [0]:

```
# DeepLabV3+ model

class DeepLabV3plus(Model):


    def __init__(self, input_size=(512, 512, 3), n_classes=4):
        super(DeepLabV3plus, self).__init__()

        self.n_classes = n_classes
        self.input_size = input_size

        # Encoder block
        self.conv2d1 = layers.Conv2D(32, (3, 3), strides=2, name='entry_conv1', padding='same')
        self.bn1 = layers.BatchNormalization(name='entry_BN')
        self.custom_conv1 = Conv2D_custom(64, kernel_size=3, stride=1, prefix='entry_conv2')
        self.bn2 = layers.BatchNormalization(name='conv2_s1_BN')
```

```
        self.bn2 = layers.BatchNormalization(name='conv2_s1_BN')

        self.entry_xception1 = Xception_Block([128, 128, 128], prefix='entry_x1', residual_type='conv', stride=2, rate=1)
        self.entry_xception2 = Xception_Block([256, 256, 256], prefix='entry_x2', residual_type='conv', stride=2, rate=1, return_skip=True)
        self.entry_xception3 = Xception_Block([728, 728, 728], prefix='entry_x3', residual_type='conv', stride=2, rate=1)

        self.middle_xception = [Xception_Block([728, 728, 728], prefix=f'middle_x{i+1}', residual_type='sum', stride=1, rate=1) for i in range(16)]

        self.exit_xception1 = Xception_Block([728, 1024, 1024], prefix='exit_x1', residual_type='conv', stride=1, rate=1)
        self.exit_xception2 = Xception_Block([1536, 1536, 2048], prefix='exit_x2', residual_type=None, stride=1, rate=2, depth_activation=True)

        # Feature projection
        self.conv_feat = layers.Conv2D(256, (1, 1), padding='same', name='conv_featureProj')
        self.bn_feat = layers.BatchNormalization(name='featureProj_BN')
        self.atrous_conv1 = SeparableConv_BN(filters=256, prefix='aspp1', stride=1, rate=6, depth_activation=True)
        self.atrous_conv2 = SeparableConv_BN(filters=256, prefix='aspp2', stride=1, rate=12, depth_activation=True)
        self.atrous_conv3 = SeparableConv_BN(filters=256, prefix='aspp3', stride=1, rate=18, depth_activation=True)
        self.image_pooling = layers.AveragePooling2D(8)
        self.conv_pool = layers.Conv2D(256, (1, 1), padding='same', name='conv_imgPool')
        self.bn_pool = layers.BatchNormalization(name='imgPool_BN')
        self.concat1 = layers.Concatenate()
        self.encoder_op = layers.Conv2D(256, (1, 1), padding='same', name='conv_encoder_op')
        self.bn_enc = layers.BatchNormalization(name='encoder_op_BN')

        # Decoder block
        self.upsample1 = layers.UpSampling2D(size=4)
        self.conv_low = layers.Conv2D(48, (1, 1), padding='same', name='conv_lowlevel_f')
        self.bn_low = layers.BatchNormalization(name='low_BN')
        self.concat2 = layers.Concatenate()
        self.sepconv_last = SeparableConv_BN(filters=256, prefix='final_sepconv', stride=1, depth_activation=True)

        self.out_conv = layers.Conv2D(self.n_classes, (1, 1), activation='sigmoid', padding='same', name='output_layer')
        self.upsample2 = layers.UpSampling2D(size=4)

    def call(self, inputs):
        #====================#
        #   Encoder Network  #
        #====================#
        # Entry Block
        x = self.conv2d1(inputs)
        x = self.bn1(x)
        x = tf.nn.relu(x)

        x = self.custom_conv1(x)
        x = self.bn2(x)

        x = self.entry_xception1(x)
        x, skip1 = self.entry_xception2(x)
        x = self.entry_xception3(x)

        # Middle Block
        for i in range(16):
            x = self.middle_xception[i](x)

        # Exit Block
        x = self.exit_xception1(x)
        x = self.exit_xception2(x)

        #=====================#
        # Feature Projection #
        #=====================#

        b0 = self.conv_feat(x)
        b0 = self.bn_feat(b0)
        b0 = tf.nn.relu(b0)

        b1 = self.atrous_conv1(x)
```

```
        b1 = self.atrous_conv1(x)
        b2 = self.atrous_conv2(x)
        b3 = self.atrous_conv3(x)

        # Image Pooling
        b4 = self.image_pooling(x)
        b4 = self.conv_pool(b4)
        b4 = self.bn_pool(b4)
        b4 = tf.nn.relu(b4)
        b4 = tf.image.resize(b4, size=[b3.get_shape()[1], b3.get_shape()[2]])


        x = self.concat1([b4, b0, b1, b2, b3])

        x = self.encoder_op(x)
        x = self.bn_enc(x)
        x = tf.nn.relu(x)
        x = tf.nn.dropout(x, rate=0.1)

        #===================#
        #  Decoder Network  #
        #===================#

        x = self.upsample1(x)

        low_level = self.conv_low(skip1)
        low_level = self.bn_low(low_level)
        low_level = tf.nn.relu(low_level)
        x = self.concat2([x, low_level])

        x = self.sepconv_last(x)

        x = self.out_conv(x)
        x = self.upsample2(x)
        return x
```

In [0]:

```
# tf.keras.backend.clear_session()
model = DeepLabV3plus(input_size=(128, 800,3))
```

In [0]:

```
batch_size = 16

train_batches = tfdata_generator(train_image_paths, train_label_paths, is_training=True,
batch_size=batch_size)
valid_batches = tfdata_generator(val_image_paths, val_label_paths, is_training=False, batch_size=ba
tch_size)
```

In [0]:

```
_ = model(train_batches.__iter__().get_next()[0]) #building the model by initializing with first
input batch
```

In [0]:

```
model.compile(optimizer=Adam(), loss=bce_dice_loss, metrics=[dice_coef])
```

In [0]:

```
model.summary()
```

```
Model: "deep_lab_v3plus"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
entry_conv1 (Conv2D)         multiple                  896
_________________________________________________________________
entry_BN (BatchNormalization multiple                  128
_________________________________________________________________
conv2d_custom (Conv2D_custom multiple                  18496
```

```
_________________________________________________________________
conv2_s1_BN (BatchNormalizat multiple                  256
_________________________________________________________________
xception__block (Xception_Bl multiple                  56192
_________________________________________________________________
xception__block_1 (Xception_ multiple                  210688
_________________________________________________________________
xception__block_2 (Xception_ multiple                  1471232
_________________________________________________________________
xception__block_3 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_4 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_5 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_6 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_7 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_8 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_9 (Xception_ multiple                  1631448
_________________________________________________________________
xception__block_10 (Xception multiple                  1631448
_________________________________________________________________
xception__block_11 (Xception multiple                  1631448
_________________________________________________________________
xception__block_12 (Xception multiple                  1631448
_________________________________________________________________
xception__block_13 (Xception multiple                  1631448
_________________________________________________________________
xception__block_14 (Xception multiple                  1631448
_________________________________________________________________
xception__block_15 (Xception multiple                  1631448
_________________________________________________________________
xception__block_16 (Xception multiple                  1631448
_________________________________________________________________
xception__block_17 (Xception multiple                  1631448
_________________________________________________________________
xception__block_18 (Xception multiple                  1631448
_________________________________________________________________
xception__block_19 (Xception multiple                  3123224
_________________________________________________________________
xception__block_20 (Xception multiple                  7160832
_________________________________________________________________
conv_featureProj (Conv2D)    multiple                  524544
_________________________________________________________________
featureProj_BN (BatchNormali multiple                  1024
_________________________________________________________________
separable_conv_bn_63 (Separa multiple                  554240
_________________________________________________________________
separable_conv_bn_64 (Separa multiple                  554240
_________________________________________________________________
separable_conv_bn_65 (Separa multiple                  554240
_________________________________________________________________
average_pooling2d (AveragePo multiple                  0
_________________________________________________________________
conv_imgPool (Conv2D)        multiple                  524544
_________________________________________________________________
imgPool_BN (BatchNormalizati multiple                  1024
_________________________________________________________________
concatenate (Concatenate)    multiple                  0
_________________________________________________________________
conv_encoder_op (Conv2D)     multiple                  327936
_________________________________________________________________
encoder_op_BN (BatchNormaliz multiple                  1024
_________________________________________________________________
up_sampling2d (UpSampling2D) multiple                  0
_________________________________________________________________
conv_lowlevel_f (Conv2D)     multiple                  12336
_________________________________________________________________
low_BN (BatchNormalization)  multiple                  192
_________________________________________________________________
concatenate_1 (Concatenate)  multiple                  0
_________________________________________________________________
separable_conv_bn_66 (Separa multiple                  83360
```

```
_________________________________________________________________
output_layer (Conv2D)        multiple                  1028      
_________________________________________________________________
up_sampling2d_1 (UpSampling2 multiple                  0         
=================================================================
Total params: 41,284,844
Trainable params: 41,083,068
Non-trainable params: 201,776
_________________________________________________________________
```

4.3. Tensorboard, Checkpoint. Creating callback list

In [0]:

```
# tensor-board in colab
# Refer: https://www.tensorflow.org/tensorboard/get_started
import os
import datetime

! rm -rf ./logs/
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
print(logdir)
```

```
logs/20200504-050629
```

In [0]:

```
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import CSVLogger

filepath="/content/drive/My Drive/CaseStudy2/"
checkpoints = ModelCheckpoint(filepath+'weights_aug_50.h5', monitor='val_dice_coef', save_weights_o
nly=True, verbose=1, save_best_only=True, mode='max')
train_log = CSVLogger(filepath+'history__aug_50.log') #storing the training results in a pandas da
taframe

tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)

callbacks_list = [checkpoints, train_log, tensorboard_callback]
```

In [0]:

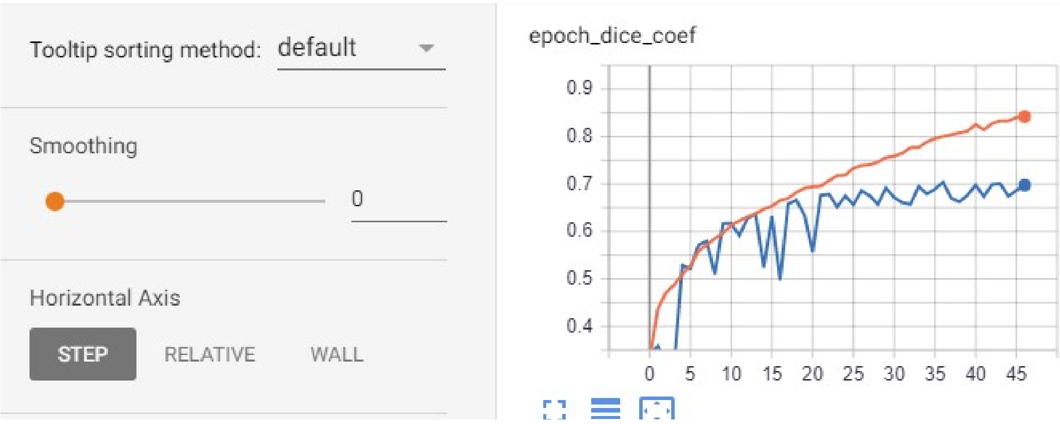
```
%load_ext tensorboard
%tensorboard --logdir $logdir
```

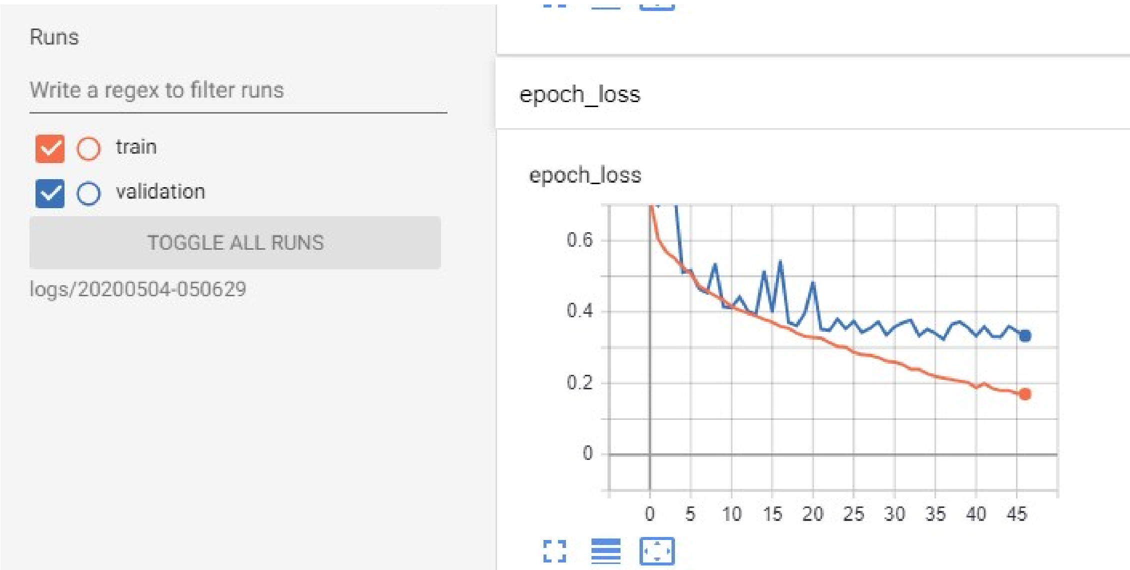
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [67]:

```
from IPython.display import Image
Image(url='https://imgur.com/akazboj.png')
```

Out[67]:





4.4. Training

In [0]:

```
history1 = model.fit_generator(train_batches, validation_data=valid_batches, epochs=50, verbose=1,
callbacks=callbacks_list)
```

```
Epoch 1/50
668/668 [==============================] - ETA: 0s - loss: 0.7220 - dice_coef: 0.3300
Epoch 00001: val_dice_coef improved from -inf to 0.34146, saving model to /content/drive/My
Drive/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 753s 1s/step - loss: 0.7220 - dice_coef: 0.3300 - val_l
oss: 0.7213 - val_dice_coef: 0.3415
Epoch 2/50
668/668 [==============================] - ETA: 0s - loss: 0.6043 - dice_coef: 0.4372
Epoch 00002: val_dice_coef improved from 0.34146 to 0.35881, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 741s 1s/step - loss: 0.6043 - dice_coef: 0.4372 - val_l
oss: 0.6992 - val_dice_coef: 0.3588
Epoch 3/50
668/668 [==============================] - ETA: 0s - loss: 0.5680 - dice_coef: 0.4714
Epoch 00003: val_dice_coef did not improve from 0.35881
668/668 [==============================] - 733s 1s/step - loss: 0.5680 - dice_coef: 0.4714 - val_l
oss: 0.7511 - val_dice_coef: 0.3230
Epoch 4/50
668/668 [==============================] - ETA: 0s - loss: 0.5506 - dice_coef: 0.4874
Epoch 00004: val_dice_coef did not improve from 0.35881
668/668 [==============================] - 730s 1s/step - loss: 0.5506 - dice_coef: 0.4874 - val_l
oss: 0.7406 - val_dice_coef: 0.3174
Epoch 5/50
668/668 [==============================] - ETA: 0s - loss: 0.5265 - dice_coef: 0.5096
Epoch 00005: val_dice_coef improved from 0.35881 to 0.52839, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 732s 1s/step - loss: 0.5265 - dice_coef: 0.5096 - val_l
oss: 0.5102 - val_dice_coef: 0.5284
Epoch 6/50
668/668 [==============================] - ETA: 0s - loss: 0.5064 - dice_coef: 0.5279
Epoch 00006: val_dice_coef did not improve from 0.52839
668/668 [==============================] - 729s 1s/step - loss: 0.5064 - dice_coef: 0.5279 - val_l
oss: 0.5167 - val_dice_coef: 0.5218
Epoch 7/50
668/668 [==============================] - ETA: 0s - loss: 0.4733 - dice_coef: 0.5590
Epoch 00007: val_dice_coef improved from 0.52839 to 0.57177, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 729s 1s/step - loss: 0.4733 - dice_coef: 0.5590 - val_l
oss: 0.4650 - val_dice_coef: 0.5718
Epoch 8/50
668/668 [==============================] - ETA: 0s - loss: 0.4593 - dice_coef: 0.5717
Epoch 00008: val_dice_coef improved from 0.57177 to 0.57949, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 735s 1s/step - loss: 0.4593 - dice_coef: 0.5717 - val_l
oss: 0.4539 - val_dice_coef: 0.5795
```

```
Epoch 9/50
668/668 [==============================] - ETA: 0s - loss: 0.4451 - dice_coef: 0.5855
Epoch 00009: val_dice_coef did not improve from 0.57949
668/668 [==============================] - 784s 1s/step - loss: 0.4451 - dice_coef: 0.5855 - val_l
oss: 0.5358 - val_dice_coef: 0.5097
Epoch 10/50
668/668 [==============================] - ETA: 0s - loss: 0.4338 - dice_coef: 0.5962
Epoch 00010: val_dice_coef improved from 0.57949 to 0.61660, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 740s 1s/step - loss: 0.4338 - dice_coef: 0.5962 - val_l
oss: 0.4148 - val_dice_coef: 0.6166
Epoch 11/50
668/668 [==============================] - ETA: 0s - loss: 0.4152 - dice_coef: 0.6132
Epoch 00011: val_dice_coef improved from 0.61660 to 0.61724, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 735s 1s/step - loss: 0.4152 - dice_coef: 0.6132 - val_l
oss: 0.4116 - val_dice_coef: 0.6172
Epoch 12/50
668/668 [==============================] - ETA: 0s - loss: 0.4052 - dice_coef: 0.6227
Epoch 00012: val_dice_coef did not improve from 0.61724
668/668 [==============================] - 736s 1s/step - loss: 0.4052 - dice_coef: 0.6227 - val_l
oss: 0.4422 - val_dice_coef: 0.5917
Epoch 13/50
668/668 [==============================] - ETA: 0s - loss: 0.3967 - dice_coef: 0.6308
Epoch 00013: val_dice_coef improved from 0.61724 to 0.62792, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 735s 1s/step - loss: 0.3967 - dice_coef: 0.6308 - val_l
oss: 0.4028 - val_dice_coef: 0.6279
Epoch 14/50
668/668 [==============================] - ETA: 0s - loss: 0.3890 - dice_coef: 0.6379
Epoch 00014: val_dice_coef improved from 0.62792 to 0.63698, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 734s 1s/step - loss: 0.3890 - dice_coef: 0.6379 - val_l
oss: 0.3927 - val_dice_coef: 0.6370
Epoch 15/50
668/668 [==============================] - ETA: 0s - loss: 0.3789 - dice_coef: 0.6470
Epoch 00015: val_dice_coef did not improve from 0.63698
668/668 [==============================] - 731s 1s/step - loss: 0.3789 - dice_coef: 0.6470 - val_l
oss: 0.5156 - val_dice_coef: 0.5238
Epoch 16/50
668/668 [==============================] - ETA: 0s - loss: 0.3715 - dice_coef: 0.6537
Epoch 00016: val_dice_coef did not improve from 0.63698
668/668 [==============================] - 732s 1s/step - loss: 0.3715 - dice_coef: 0.6537 - val_l
oss: 0.3989 - val_dice_coef: 0.6324
Epoch 17/50
668/668 [==============================] - ETA: 0s - loss: 0.3596 - dice_coef: 0.6653
Epoch 00017: val_dice_coef did not improve from 0.63698
668/668 [==============================] - 730s 1s/step - loss: 0.3596 - dice_coef: 0.6653 - val_l
oss: 0.5444 - val_dice_coef: 0.4968
Epoch 18/50
668/668 [==============================] - ETA: 0s - loss: 0.3541 - dice_coef: 0.6698
Epoch 00018: val_dice_coef improved from 0.63698 to 0.65828, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 732s 1s/step - loss: 0.3541 - dice_coef: 0.6698 - val_l
oss: 0.3709 - val_dice_coef: 0.6583
Epoch 19/50
668/668 [==============================] - ETA: 0s - loss: 0.3409 - dice_coef: 0.6823
Epoch 00019: val_dice_coef improved from 0.65828 to 0.66606, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 734s 1s/step - loss: 0.3409 - dice_coef: 0.6823 - val_l
oss: 0.3611 - val_dice_coef: 0.6661
Epoch 20/50
668/668 [==============================] - ETA: 0s - loss: 0.3313 - dice_coef: 0.6914
Epoch 00020: val_dice_coef did not improve from 0.66606
668/668 [==============================] - 737s 1s/step - loss: 0.3313 - dice_coef: 0.6914 - val_l
oss: 0.3971 - val_dice_coef: 0.6346
Epoch 21/50
668/668 [==============================] - ETA: 0s - loss: 0.3286 - dice_coef: 0.6940
Epoch 00021: val_dice_coef did not improve from 0.66606
668/668 [==============================] - 733s 1s/step - loss: 0.3286 - dice_coef: 0.6940 - val_l
oss: 0.4843 - val_dice_coef: 0.5567
Epoch 22/50
668/668 [==============================] - ETA: 0s - loss: 0.3266 - dice_coef: 0.6956
Epoch 00022: val_dice_coef improved from 0.66606 to 0.67626, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 733s 1s/step - loss: 0.3266 - dice_coef: 0.6956 - val_l
oss: 0.3515 - val_dice_coef: 0.6763
```

```
Epoch 23/50
668/668 [==============================] - ETA: 0s - loss: 0.3142 - dice_coef: 0.7073
Epoch 00023: val_dice_coef improved from 0.67626 to 0.67853, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 733s 1s/step - loss: 0.3142 - dice_coef: 0.7073 - val_l
oss: 0.3484 - val_dice_coef: 0.6785
Epoch 24/50
668/668 [==============================] - ETA: 0s - loss: 0.3032 - dice_coef: 0.7178
Epoch 00024: val_dice_coef did not improve from 0.67853
668/668 [==============================] - 732s 1s/step - loss: 0.3032 - dice_coef: 0.7178 - val_l
oss: 0.3796 - val_dice_coef: 0.6521
Epoch 25/50
668/668 [==============================] - ETA: 0s - loss: 0.3013 - dice_coef: 0.7191
Epoch 00025: val_dice_coef did not improve from 0.67853
668/668 [==============================] - 735s 1s/step - loss: 0.3013 - dice_coef: 0.7191 - val_l
oss: 0.3532 - val_dice_coef: 0.6745
Epoch 26/50
668/668 [==============================] - ETA: 0s - loss: 0.2864 - dice_coef: 0.7334
Epoch 00026: val_dice_coef did not improve from 0.67853
668/668 [==============================] - 733s 1s/step - loss: 0.2864 - dice_coef: 0.7334 - val_l
oss: 0.3734 - val_dice_coef: 0.6570
Epoch 27/50
668/668 [==============================] - ETA: 0s - loss: 0.2803 - dice_coef: 0.7392
Epoch 00027: val_dice_coef improved from 0.67853 to 0.68578, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 733s 1s/step - loss: 0.2803 - dice_coef: 0.7392 - val_l
oss: 0.3428 - val_dice_coef: 0.6858
Epoch 28/50
668/668 [==============================] - ETA: 0s - loss: 0.2782 - dice_coef: 0.7409
Epoch 00028: val_dice_coef did not improve from 0.68578
668/668 [==============================] - 729s 1s/step - loss: 0.2782 - dice_coef: 0.7409 - val_l
oss: 0.3545 - val_dice_coef: 0.6752
Epoch 29/50
668/668 [==============================] - ETA: 0s - loss: 0.2723 - dice_coef: 0.7461
Epoch 00029: val_dice_coef did not improve from 0.68578
668/668 [==============================] - 728s 1s/step - loss: 0.2723 - dice_coef: 0.7461 - val_l
oss: 0.3718 - val_dice_coef: 0.6578
Epoch 30/50
668/668 [==============================] - ETA: 0s - loss: 0.2619 - dice_coef: 0.7561
Epoch 00030: val_dice_coef improved from 0.68578 to 0.69143, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 736s 1s/step - loss: 0.2619 - dice_coef: 0.7561 - val_l
oss: 0.3358 - val_dice_coef: 0.6914
Epoch 31/50
668/668 [==============================] - ETA: 0s - loss: 0.2593 - dice_coef: 0.7587
Epoch 00031: val_dice_coef did not improve from 0.69143
668/668 [==============================] - 736s 1s/step - loss: 0.2593 - dice_coef: 0.7587 - val_l
oss: 0.3581 - val_dice_coef: 0.6715
Epoch 32/50
668/668 [==============================] - ETA: 0s - loss: 0.2526 - dice_coef: 0.7649
Epoch 00032: val_dice_coef did not improve from 0.69143
668/668 [==============================] - 730s 1s/step - loss: 0.2526 - dice_coef: 0.7649 - val_l
oss: 0.3698 - val_dice_coef: 0.6609
Epoch 33/50
668/668 [==============================] - ETA: 0s - loss: 0.2397 - dice_coef: 0.7768
Epoch 00033: val_dice_coef did not improve from 0.69143
668/668 [==============================] - 731s 1s/step - loss: 0.2397 - dice_coef: 0.7768 - val_l
oss: 0.3768 - val_dice_coef: 0.6574
Epoch 34/50
668/668 [==============================] - ETA: 0s - loss: 0.2393 - dice_coef: 0.7771
Epoch 00034: val_dice_coef improved from 0.69143 to 0.69513, saving model to /content/drive/My Dri
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 733s 1s/step - loss: 0.2393 - dice_coef: 0.7771 - val_l
oss: 0.3328 - val_dice_coef: 0.6951
Epoch 35/50
668/668 [==============================] - ETA: 0s - loss: 0.2270 - dice_coef: 0.7887
Epoch 00035: val_dice_coef did not improve from 0.69513
668/668 [==============================] - 729s 1s/step - loss: 0.2270 - dice_coef: 0.7887 - val_l
oss: 0.3518 - val_dice_coef: 0.6790
Epoch 36/50
668/668 [==============================] - ETA: 0s - loss: 0.2194 - dice_coef: 0.7959
Epoch 00036: val_dice_coef did not improve from 0.69513
668/668 [==============================] - 730s 1s/step - loss: 0.2194 - dice_coef: 0.7959 - val_l
oss: 0.3398 - val_dice_coef: 0.6892
Epoch 37/50
668/668 [==============================] - ETA: 0s - loss: 0.2145 - dice_coef: 0.8003
Epoch 00037: val_dice_coef improved from 0.69513 to 0.70388, saving model to /content/drive/My Dri
```

```
ve/CaseStudy2/weights_aug_50.h5
668/668 [==============================] - 732s 1s/step - loss: 0.2145 - dice_coef: 0.8003 - val_l
oss: 0.3233 - val_dice_coef: 0.7039
Epoch 38/50
668/668 [==============================] - ETA: 0s - loss: 0.2107 - dice_coef: 0.8039
Epoch 00038: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 730s 1s/step - loss: 0.2107 - dice_coef: 0.8039 - val_l
oss: 0.3649 - val_dice_coef: 0.6701
Epoch 39/50
668/668 [==============================] - ETA: 0s - loss: 0.2062 - dice_coef: 0.8082
Epoch 00039: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 727s 1s/step - loss: 0.2062 - dice_coef: 0.8082 - val_l
oss: 0.3724 - val_dice_coef: 0.6624
Epoch 40/50
668/668 [==============================] - ETA: 0s - loss: 0.2025 - dice_coef: 0.8116
Epoch 00040: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 727s 1s/step - loss: 0.2025 - dice_coef: 0.8116 - val_l
oss: 0.3560 - val_dice_coef: 0.6766
Epoch 41/50
668/668 [==============================] - ETA: 0s - loss: 0.1878 - dice_coef: 0.8254
Epoch 00041: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 728s 1s/step - loss: 0.1878 - dice_coef: 0.8254 - val_l
oss: 0.3322 - val_dice_coef: 0.6979
Epoch 42/50
668/668 [==============================] - ETA: 0s - loss: 0.1996 - dice_coef: 0.8143
Epoch 00042: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 729s 1s/step - loss: 0.1996 - dice_coef: 0.8143 - val_l
oss: 0.3589 - val_dice_coef: 0.6734
Epoch 43/50
668/668 [==============================] - ETA: 0s - loss: 0.1858 - dice_coef: 0.8272
Epoch 00043: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 726s 1s/step - loss: 0.1858 - dice_coef: 0.8272 - val_l
oss: 0.3309 - val_dice_coef: 0.6988
Epoch 44/50
668/668 [==============================] - ETA: 0s - loss: 0.1794 - dice_coef: 0.8332
Epoch 00044: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 726s 1s/step - loss: 0.1794 - dice_coef: 0.8332 - val_l
oss: 0.3296 - val_dice_coef: 0.7008
Epoch 45/50
668/668 [==============================] - ETA: 0s - loss: 0.1793 - dice_coef: 0.8331
Epoch 00045: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 730s 1s/step - loss: 0.1793 - dice_coef: 0.8331 - val_l
oss: 0.3597 - val_dice_coef: 0.6744
Epoch 46/50
668/668 [==============================] - ETA: 0s - loss: 0.1721 - dice_coef: 0.8400
Epoch 00046: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 733s 1s/step - loss: 0.1721 - dice_coef: 0.8400 - val_l
oss: 0.3458 - val_dice_coef: 0.6865
Epoch 47/50
668/668 [==============================] - ETA: 0s - loss: 0.1696 - dice_coef: 0.8421
Epoch 00047: val_dice_coef did not improve from 0.70388
668/668 [==============================] - 737s 1s/step - loss: 0.1696 - dice_coef: 0.8421 - val_l
oss: 0.3331 - val_dice_coef: 0.6977
Epoch 48/50
458/668 [===================>..........] - ETA: 3:39 - loss: 0.1597 - dice_coef: 0.8518
```

Observations

- The model starts overfittingafter 37th epoch (Train score keeps improving but Validation score does not increase)
- Therefore, we have used the model weights saved at 37th epoch.

4.5. Loading the saved model and testing

In [0]:

```
model.load_weights('/content/drive/My Drive/CaseStudy2/weights_aug_50.h5')
```

In [0]:

```
evals= model.evaluate(valid_batches,verbose=1)
print('Validation score:')
print('loss:',evals[0])
print('dice_coeff:',evals[1])
```

```
print('dice_coeff: ',evals[1])
```

```
118/118 [==============================] - 33s 282ms/step - loss: 0.3236 - dice_coef: 0.7036
Validation score:
loss: 0.3236275315284729
dice_coeff: 0.7036087512969971
```

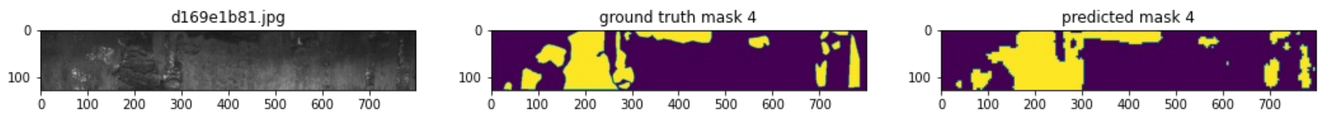
Utility function to visualize ground truth and predicted mask of an image(train/cv)

In [0]:

```
def visualize_prediction(f: str):

    data_path = 'train_images/'
    X = np.empty((1,128,800,3),dtype='float32')
    img_str = tf.io.read_file(data_path + f)
    img = tf.image.decode_jpeg(img_str, channels=3)
    img = tf.image.convert_image_dtype(img, tf.float32)
    img = tf.image.resize(img, [128, 800])
    X[0,] = img.numpy()
    mask = model.predict(X)

    df1 = data[data.ImageId == f].reset_index()

    if all((df1['Defect_1'].all()=='',df1['Defect_2'].all()=='',df1['Defect_3'].all()=='',df1['Defect_4'].all()=='')):

        fig, (ax1,ax2, ax3) = plt.subplots(nrows = 1,ncols = 3,figsize=(18, 7))
        ax1.imshow(img)
        ax1.set_title(f)

        ax2.imshow(rle2mask(''))
        ax2.set_title('ground truth mask 0')

        ax3.imshow(mask[0,:,:,0].round().astype('int'))
        ax3.set_title('predicted mask 0')
        plt.show()

    else:
        for k in [1,2,3,4]:
            for i in range(len(df1)):
                if df1[f'Defect_{k}'][i] != '':
                    encoded_pix = df1[f'Defect_{k}'][i]

                    fig, (ax1,ax2, ax3) = plt.subplots(nrows = 1,ncols = 3,figsize=(18, 7))
                    ax1.imshow(img)
                    ax1.set_title(f)

                    ax2.imshow(rle2mask(encoded_pix))
                    ax2.set_title('ground truth mask '+str(k))

                    ax3.imshow(mask[0,:,:,k-1].round().astype('int'))
                    ax3.set_title('predicted mask '+str(k))
                    plt.show()
```

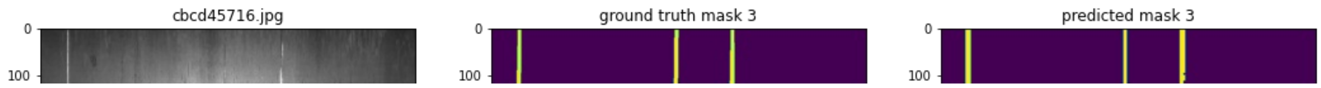
In [28]:

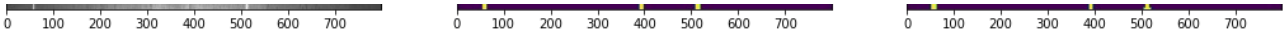
```
visualize_prediction('d169e1b81.jpg')
```



In [29]:

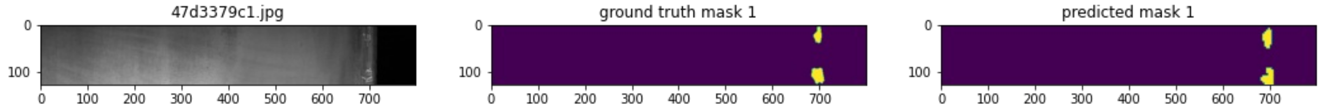
```
visualize_prediction('cbcd45716.jpg')
```





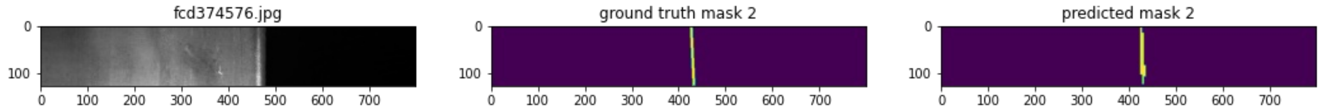
In [60]:

```python
visualize_prediction('47d3379c1.jpg')
```



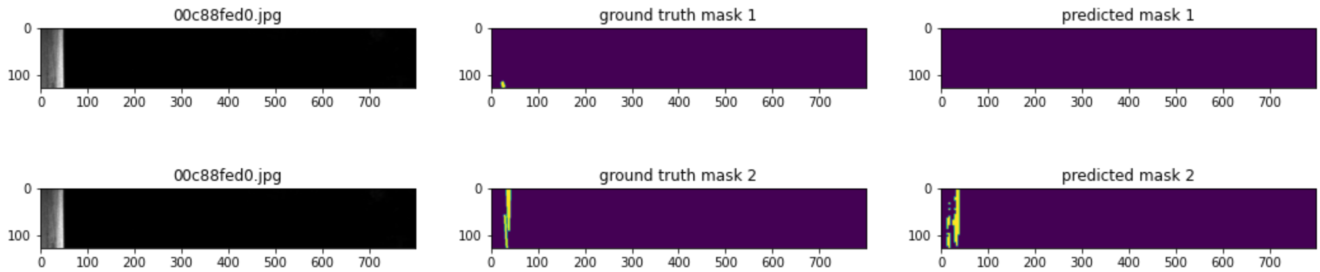
In [55]:

```python
visualize_prediction('fcd374576.jpg')
```



In [40]:

```python
visualize_prediction('00c88fed0.jpg')
```



4.6. Predict on test images

In [31]:

```python
# Predicting on test data

data_path = 'test_images/'
files = list(os.listdir(data_path))
img_ID= []
classId = []
rle_lst = []
img_classId= []
for f in tqdm(files):
    X = np.empty((1,128,800,3),dtype='float32')
    img_str = tf.io.read_file(data_path + f)
    img = tf.image.decode_jpeg(img_str, channels=3)
    img = tf.image.convert_image_dtype(img, tf.float32)
    img = tf.image.resize(img, [128, 800])
    X[0,] = img.numpy()

    mask = model.predict(X)
    mask = tf.image.resize(mask, [256, 1600]).numpy()
    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_ID.append(f)
        classId.append(str(i+1))
        img_classId.append(f+'_'+str(i+1))
```

4.7. Creating csv file for Kaggle submission

In [0]:

```
output = {'ImageId_ClassId':img_classId, 'EncodedPixels' : rle_lst}

filepath= "/content/drive/My Drive/CaseStudy2/"
output_df = pd.DataFrame(output)
output_df.to_csv(filepath+'submission_DLV3p_aug.csv', index=False)
```

5. Conclusions

- The model performs very well in localizing type1, type3 and type4 defects.
- Its performance is slighty degraded when it comes to localization of type2 defects. Nonetheless, it does brilliant job in classifying the defects accurately.
- The model performs better on the images containing single type of defects as compared to the images having multiple types of defects.
- The overall perfomance of the model is very good.
  After, submitting the predictions to Kaggle, it gave a **test dice coefficient ≈ 0.84**(in both private and public LB), which is much better than the score of my first cut solution (basic UNet), 0.805.

Steps followed for this Case Study

1. Read and understand the Business problem, map it to Machine Learning problem and identify business objectives and constraints.
2. Download the data and store in a workable format.
3. Perform basic EDA, like number of images in each class, number of classes per image, etc.
4. Define utility functions to convert run length encoded pixels to mask images and vice-versa. Also define loss and score functions (dice_loss and dice_coefficient)
5. Split the data into train and validation randomly.
6. Prepare the data for training, i.e. define image data genrators to get batches while training (used tensorflow.data.dataset).
7. Define the model architecture.

> - I first used a basic UNet acrchitecture, the code for which is in a separate notebook.
> - Later I used Google's research paper [Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation](DeepLabV3plus).
>   *I used sub-classes of tensorflow.keras.Model to define custom layers and blocks of the DeepLabV3plus model.

8. Compile the model and train with an approporiate optimizer(used Adam). Checkpoint the model weights at each epoch where validation score improves, so that you don't loose your model in case training is interrupted unintentionally.
9. Load the weights from best epoch and test the model on validation data. Visualize your predictions and see if the model is working as intended.
10. Predict on test images, convert the masks to rle pixels and save the predicitions to submissions.csv file.
11. Submit the predictions to Kaggle.