Capstone Project 3 – Steel Defects Classification and Segmentation

Background and Problem Statement:

Artificial Intelligence and Machine Learning are transforming the manufacturing industry. According to the report released by World Economic Forum, these technologies will play significant roles in the fourth industrial revolution.

Major areas which can be benefited from this are:

- Maintenance Department
- Production Department
- Supply Chain Department
- Quality Control Department
- Industrial Safety Department

Deep learning has been proven to be superior in detecting and localizing defects using imagery data which could significantly improve the production efficiency in the manufacturing industry

Detecting defects would help in improving the quality of manufacturing as well as in reducing the waste due to production defects.

Aim and Data Source:

https://www.kaggle.com/c/severstal-steel-defect-detection/data

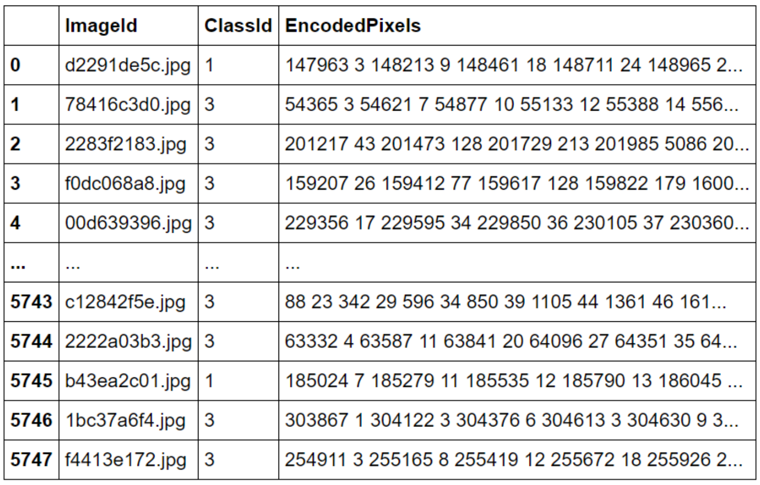
Data contains around 12600 images with 4 classes of defects, we aim to detect:

- Detect Whether the image has a defect or not.
- Classifying types of defects and localizing them with the help of a mask.

Feature Engineering & EDA:

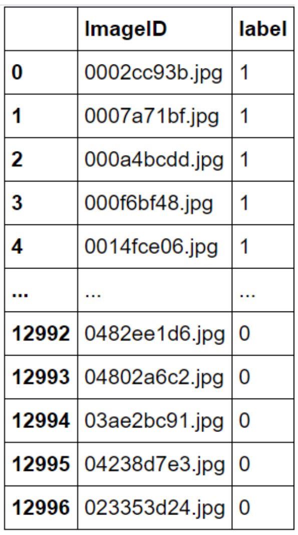
We were given two files namely

- Defective images training data containing defective images name encoded mask pixels and classes of defects.



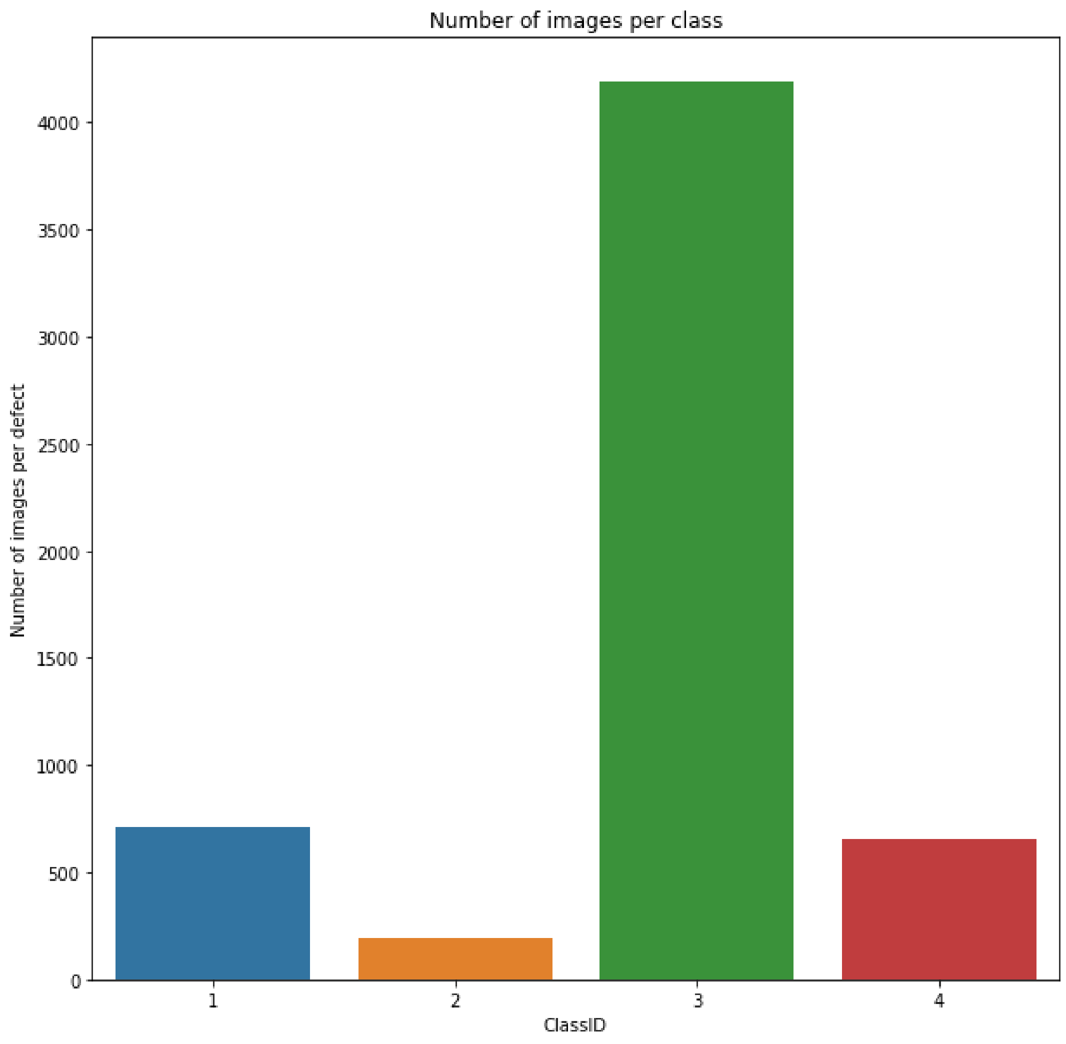
	ImageId	ClassId	EncodedPixels
0	d2291de5c.jpg	1	147963 3 148213 9 148461 18 148711 24 148965 2...
1	78416c3d0.jpg	3	54365 3 54621 7 54877 10 55133 12 55388 14 556...
2	2283f2183.jpg	3	201217 43 201473 128 201729 213 201985 5086 20...
3	f0dc068a8.jpg	3	159207 26 159412 77 159617 128 159822 179 1600...
4	00d639396.jpg	3	229356 17 229595 34 229850 36 230105 37 230360...
...	...	...	...
5743	c12842f5e.jpg	3	88 23 342 29 596 34 850 39 1105 44 1361 46 161...
5744	2222a03b3.jpg	3	63332 4 63587 11 63841 20 64096 27 64351 35 64...
5745	b43ea2c01.jpg	1	185024 7 185279 11 185535 12 185790 13 186045 ...
5746	1bc37a6f4.jpg	3	303867 1 304122 3 304376 6 304613 3 304630 9 3...
5747	f4413e172.jpg	3	254911 3 255165 8 255419 12 255672 18 255926 2...

- All image’s data both defective and non-defective along with their labels.

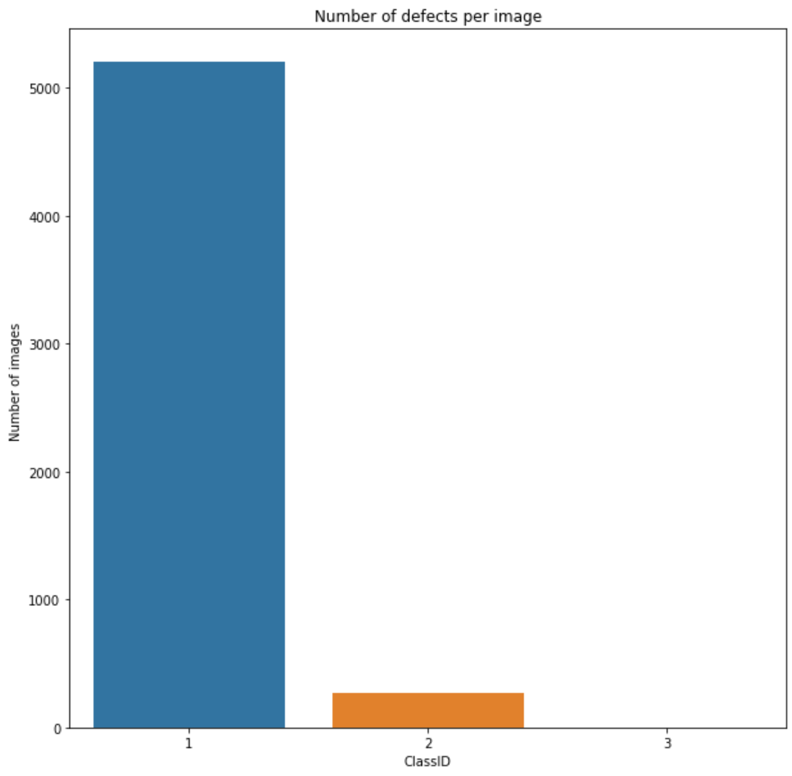


	ImageID	label
0	0002cc93b.jpg	1
1	0007a71bf.jpg	1
2	000a4bcdd.jpg	1
3	000f6bf48.jpg	1
4	0014fce06.jpg	1
...	...	...
12992	0482ee1d6.jpg	0
12993	04802a6c2.jpg	0
12994	03ae2bc91.jpg	0
12995	04238d7e3.jpg	0
12996	023353d24.jpg	0

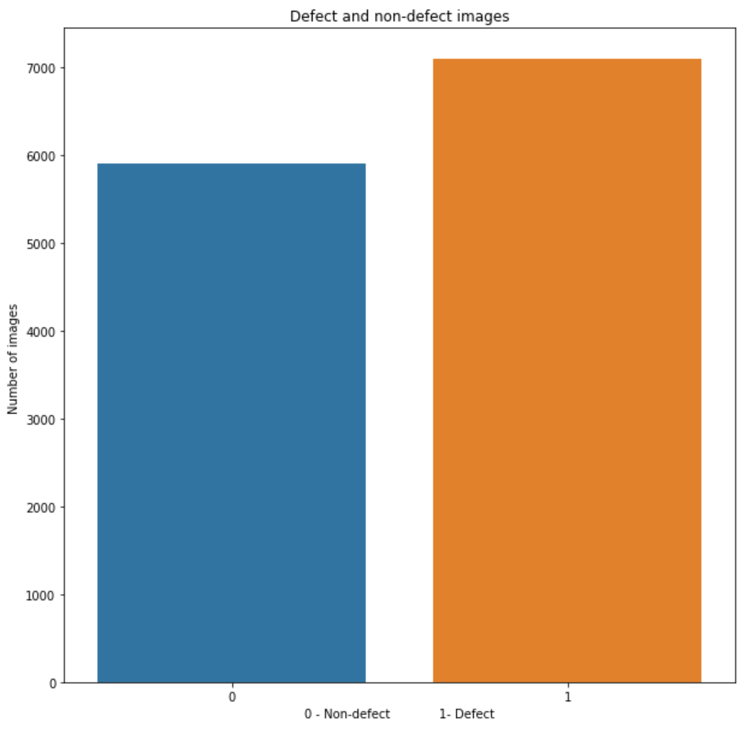
- Visualizing the Defective images dataset, we find out there is a class imbalance in the dataset with a prominent class being as a defect with class 3.



- Also, we would like to see that whether our images are having multiple defects or not plotting and doing the value counts we come to know that there are
    1. 5201 images having only 1 defect
    2. 272 images having 2 defects
    3. 1 image having all the defects.



- Analysing All images file to find out no of images having defects



- There are around 7000 images with defects and nearly 5400 with no defects.

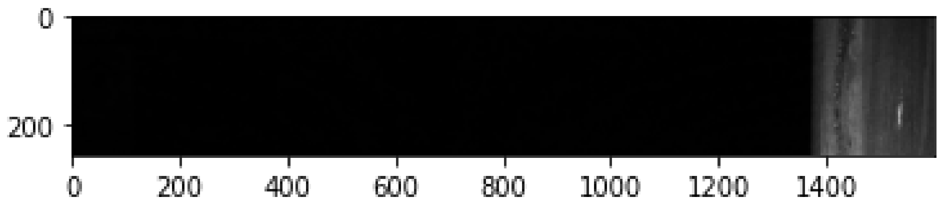
- For visualization of Defective images given the mask encoded pixels we have used run length encoding method to convert encoded pixels to image mask.

Run length Encoding Theory:

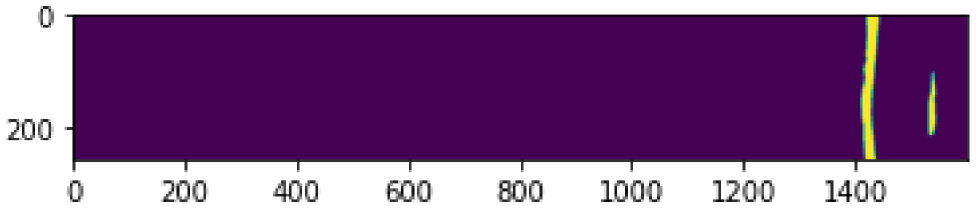
- Sometimes it is hard to represent a mask using an index as it would make the length of the mask equal to the product of height and width of the image
- To overcome this, we use a lossless data compression technique called Run-length encoding (RLE), which stores sequences that contain many consecutive data elements as a single data value followed by the count.
- For example, assume we have an image (single row) containing plain black text on a solid white background. B represents a black pixel and W represents white:

WWWWWWWWWWWWBWWWWWWWWWWWW
WBBBWWWWWWWWWWWWWWWWWWWWWW
WWWWBWWWWWWWWWWWWWW

- Its run length encoding is 12W1B12W3B24W1B14W
- This can be interpreted as a sequence of twelve Ws, one B, twelve Ws, three Bs, etc.

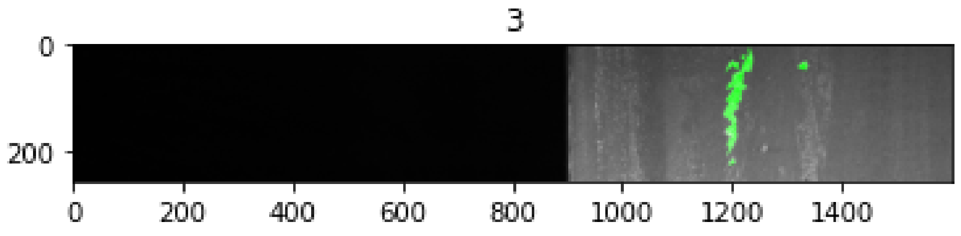


Before RLE encoding



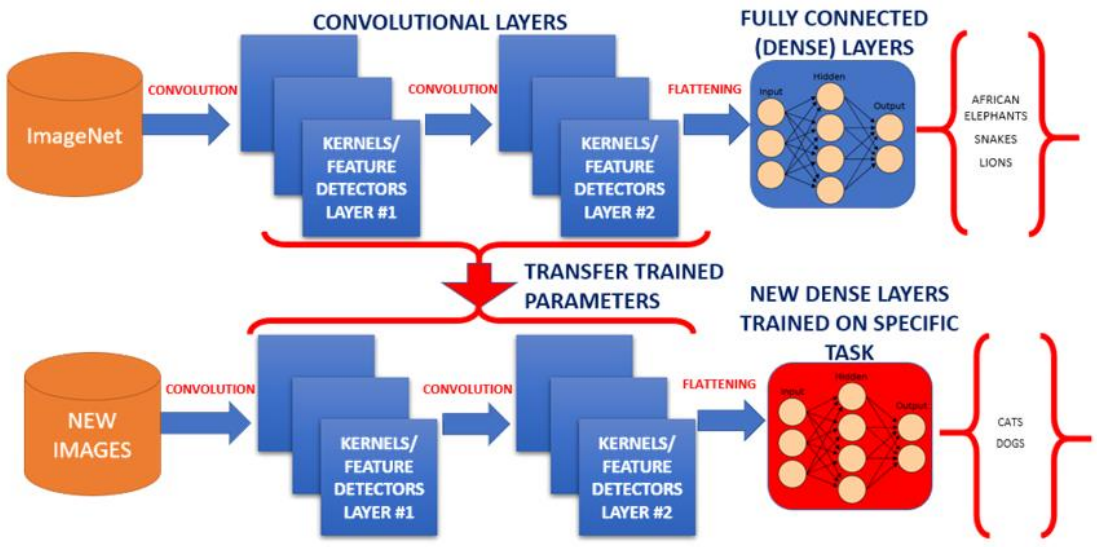
After RLE encoding

- As seen above we can now visualize the defects more clearly.
- We can further convert the image into RGB format using open cv.



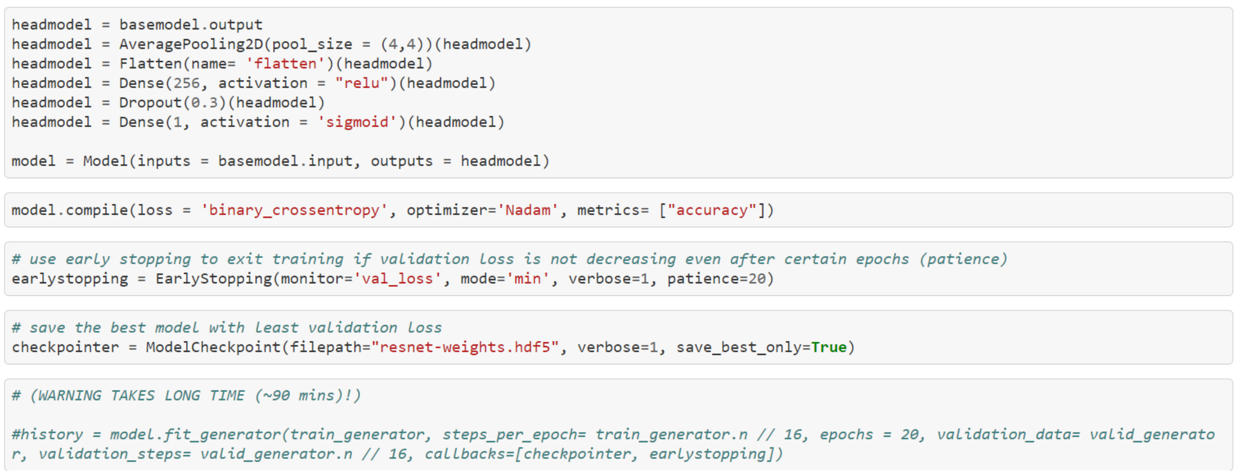
Model Preparation and Comparison:

- In the first step, we need to classify images into whether it's defective or not for this we have split the data into test and train with a ratio as 0.15 to 0.85.
- Also, we have used image augmentation by using an image data generator further splitting the data into training and validation datasets.
- Image augmentation was also performed for the test dataset.
- We have used base model as **ResNet** and have used the concept of transfer learning by using the pre calculated weights for Resnet model.


CONVOLUTIONAL LAYERS
FULLY CONNECTED (DENSE) LAYERS
ImageNet
CONVOLUTION
CONVOLUTION
FLATTENING
KERNELS/ FEATURE DETECTORS LAYER #1
KERNELS/ FEATURE DETECTORS LAYER #2
Input
Hidden
Output
AFRICAN ELEPHANTS
SNAKES
LIONS
TRANSFER TRAINED PARAMETERS
NEW DENSE LAYERS TRAINED ON SPECIFIC TASK
NEW IMAGES
CONVOLUTION
CONVOLUTION
FLATTENING
KERNELS/ FEATURE DETECTORS LAYER #1
KERNELS/ FEATURE DETECTORS LAYER #2
Input
Hidden
Output
CATS
DOGS

Overview of Transfer Learning

Model summary:



```
headmodel = basemodel.output
headmodel = AveragePooling2D(pool_size = (4,4))(headmodel)
headmodel = Flatten(name= 'flatten')(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(1, activation = 'sigmoid')(headmodel)

model = Model(inputs = basemodel.input, outputs = headmodel)
```

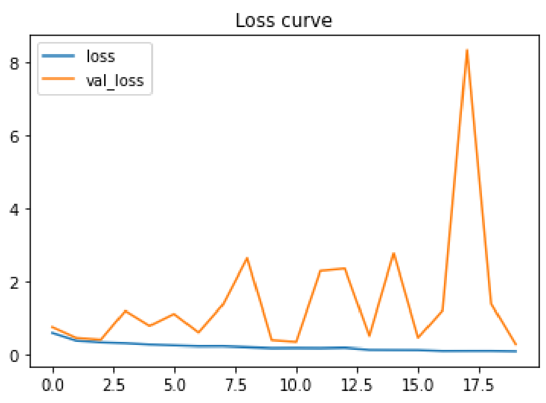
```
model.compile(loss = 'binary_crossentropy', optimizer='Nadam', metrics= ["accuracy"])
```

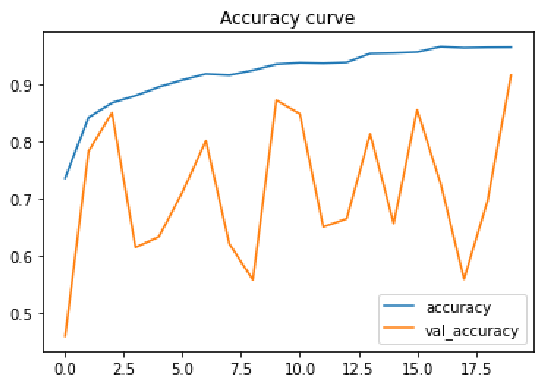
```
# use early stopping to exit training if validation loss is not decreasing even after certain epochs (patience)
earlystopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
```

```
# save the best model with least validation loss
checkpointer = ModelCheckpoint(filepath="resnet-weights.hdf5", verbose=1, save_best_only=True)
```

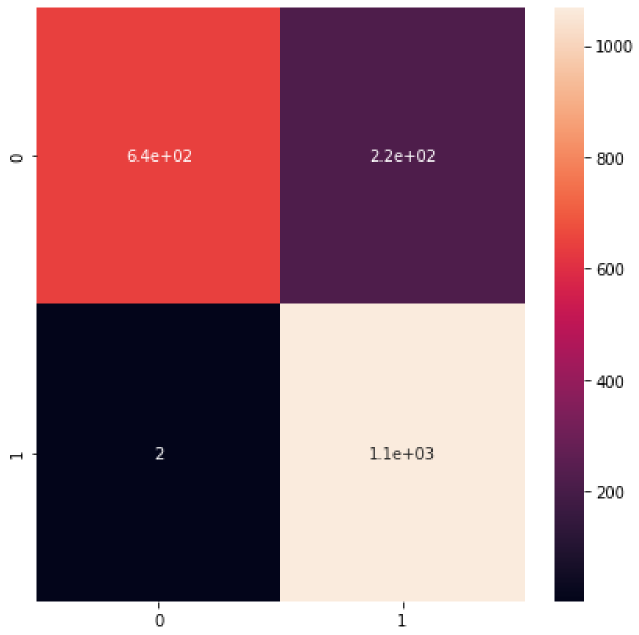
```
# (WARNING TAKES LONG TIME (~90 mins)!)

#history = model.fit_generator(train_generator, steps_per_epoch= train_generator.n // 16, epochs = 20, validation_data= valid_generato
r, validation_steps= valid_generator.n // 16, callbacks=[checkpointer, earlystopping])
```

- After compiling and fitting the model an accuracy of 96.48% on train dataset and a validation accuracy of 91.57% was observed.





- The loss curve and accuracy curve for the model can be seen above.
- For the test data model, accuracy came out to be 88.42%.
- Classification Report and confusion matrix was as follows.



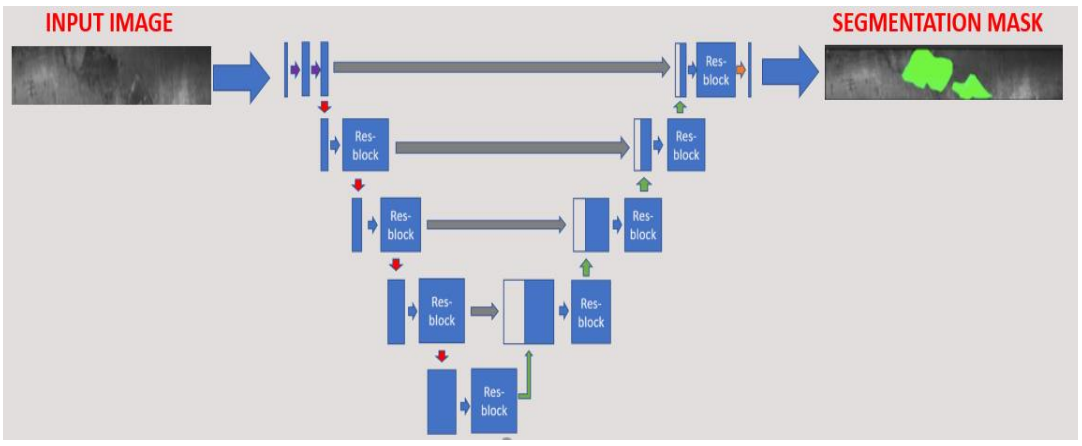
```
              precision    recall  f1-score   support

           0       1.00      0.74      0.85       866
           1       0.83      1.00      0.91      1070

    accuracy                           0.88      1936
   macro avg       0.91      0.87      0.88      1936
weighted avg       0.90      0.88      0.88      1936
```

- For our second Aim of the project to do defect segmentation we have used Res U-Net but before that less find out what is image segmentation?

What is image Segmentation?

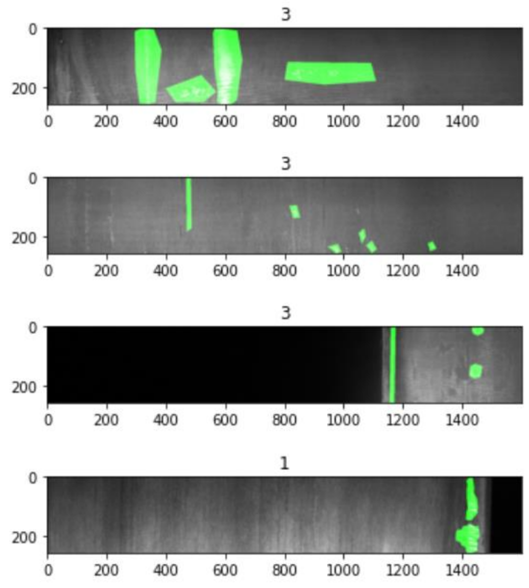
- The goal of image segmentation is to understand and extract information from images at the pixel level.
- Image Segmentation can be used for object recognition and localization which offers tremendous value in many applications such as medical imaging and self-driving cars etc.
- The goal of image segmentation is to train a neural network to produce a pixel-wise mask of the image.
- Modern image segmentation techniques are based on a deep learning approach that makes use of common architectures such as CNN, FCNs (Fully Convolution Networks), and Deep Encoders-Decoders.
- In CNN for image classification problems, we have to convert the image into a vector and possibly add a classification head at the end.
- However, in the case of Unit, we convert (encode) the image into a vector followed by up sampling (decode) it back again into an image.
- In the case of Unet, the input and output have the same size so the size of the image is preserved.
- For classical CNNs: are generally used when the entire image is needed to be classified as a class label.
- For Unet: pixel-level classification is performed.
- U-net formulates a loss function for every pixel in the input image.
- The SoftMax function is applied to every pixel which makes the segmentation problem works as a classification problem where classification is performed on every pixel of the image

Reference:
https://towardsdatascience.com/introduction-to-u-net-and-res-net-for-image-segmentation-9afcb432ee2f

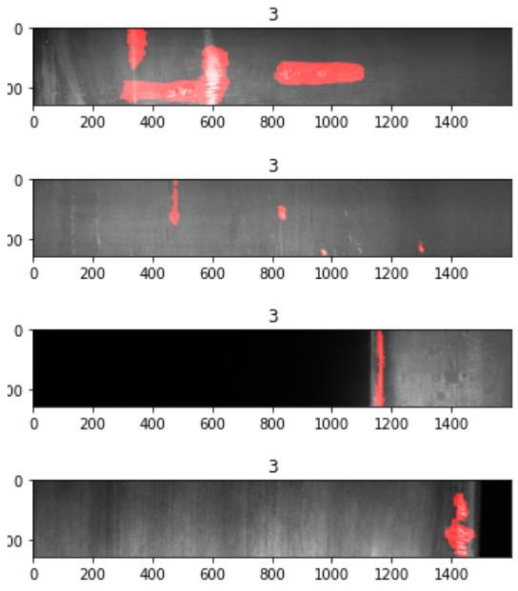


We need a custom loss function to train this ResUNet. So, we have used the loss function as it is from https://github.com/nabsabraham/focal-tversky-unet/blob/master/losses.py

Visualizing The results of Unet Segmentation Model



Train set



Test Set

As seen from above in 4th image we have misclassified the class however we were able to correctly segment the defects.

Suggestions/ Future Research:

- Our model correctly identifies defective and non-defective images with 88% accuracy.
- For the segmentation model we need to come up with some metric to compare our test and training results both segmentation and class predictions.
- Furthermore, work needs to be done how can we put the model inro production for real time testing.
- Model Retraining approach can be applied for training the model for new images or new defects.