Detection of Steel defects using deep learning techniques

Problem statement :

Semantic Segmentation of various defects on steel surfaces

Detection, classification, and segmentation of various types of steel defects present in the given images of casted steel using image processing and machine learning techniques.

Four types of steel defects namely Small Divots, Surface Scratches, Vertical Cracks, and Ablation defects have been considered.

Each image might have zero, one, or multiple defects. The pixels corresponding to the defect locations are to be estimated.

Methodology :

A 3 step pipeline :

1. Preprocess the data to fit into the model(data_pre.ipynb)
2. Train a CNN on the available data(train.ipynb)
3. Apply thresholding based on area and probabilities for each image(predict.ipynb)

CNN used :

A CNN with UNet architecture is used to predict the masks(one for each defect) of an image.

Transfer Learning :

A resnet34 pre-trained on the Imagenet dataset is used as the encoder or the backbone.

Freezing the encoder :

In order to avoid the untrained decoder from tampering with the weights on the pre-trained encoder, the encoder is kept frozen for the first 2 or 3 epochs. After the decoder learns to some extent the encoder is released and the weights on its filters can be updated based on the errors

Threshold on probabilities :

In each mask predicted by the Unet the values of pixels $\epsilon[0,1]$, which indicate the probability of those pixels having the considered defect. Thresholding is applied to the pixel values Inorder to exactly predict the defect area.

Thresholding based on area :

Even though there are no defects the model may wrongly predict some pixels as defect centers. In order to avoid such spurious estimations from affecting the metric measure,

pixels areas less than particular area threshold values are disregarded(the threshold value changes based on the defect with defect 3 having the highest threshold)
These threshold values are determined based on the defect area distributions on images in the training dataset for each defect.

Accuracy measure :

For each image img : Let Y be the actual set of pixels corresponding to a particular defect and X be the estimated set of pixels by the neural network. Then,

Accuracy of segmenting the defect area in an image img = $\frac{2\times|X\cap Y|}{|X|+|Y|}$

The overall accuracy can be calculated by taking the average over individual image accuracies.
A smoothening factor of 1 is added to avoid the denominator becoming 0.

Dataset :

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

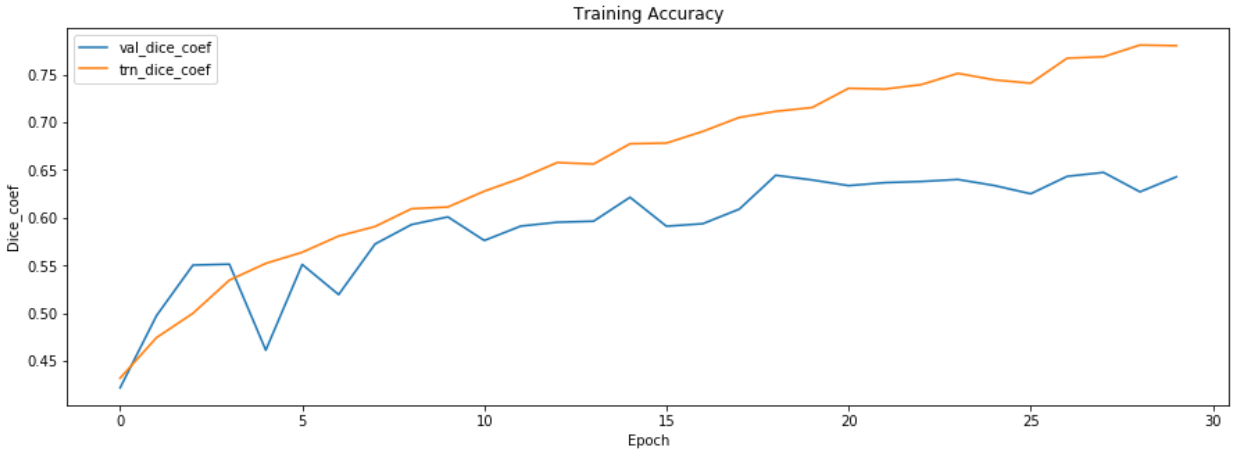
Since there are no results provided for the test data set to check the accuracy of the Unet on it, The training dataset is split into 3 parts with 60% of the data for training, 20% for validation in order to avoid overfitting, and 10%for testing.

The images are resized into smaller images(128*800), before passing them to the neural network, because of their large size(256*1600)

(The new dataset CSV file and the trained model saved are to be imported before using them in other notebooks)

Results :

The decoder is initially trained for 2 epochs keeping the encoder frozen,the complete model is trained over the dataset for 30 epochs.
The training and validation dice coefficients are as follows :



The validation dice coefficient is around 0.66 after the training. The value is low because :

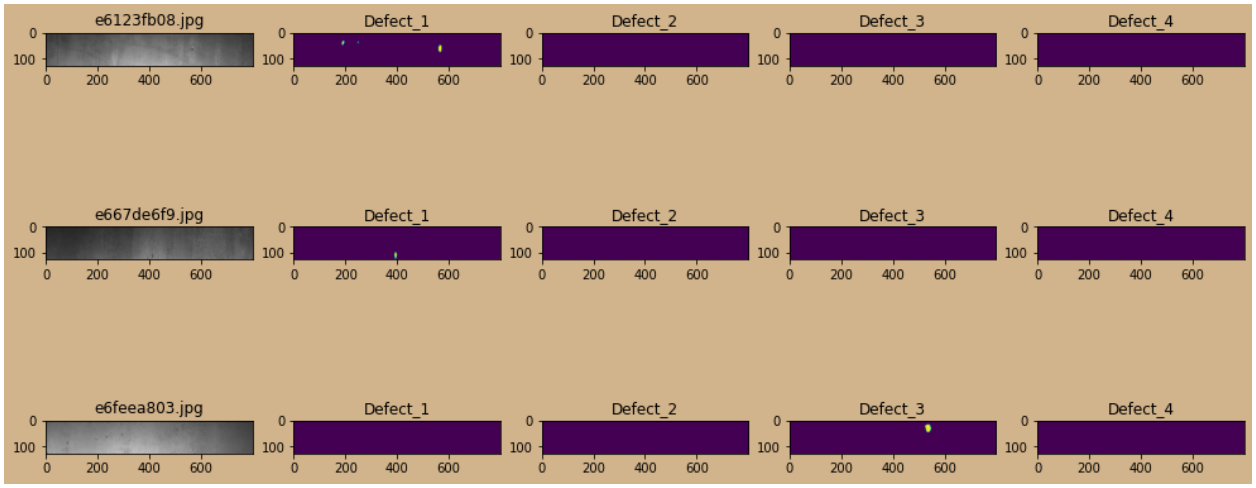
1. the masks predicted to have the confidence values which are to normalized s.t values can either be 0 or 1 instead of a range
2. The model might wrongly predict some pixels as having defects even though the particular kind of defect is not present in the considered image.In order to eliminate this we need to apply a threshold on the total defect area.

(The final values for these thresholds are taken from the EDA done in other work. The work has been cited in our references)

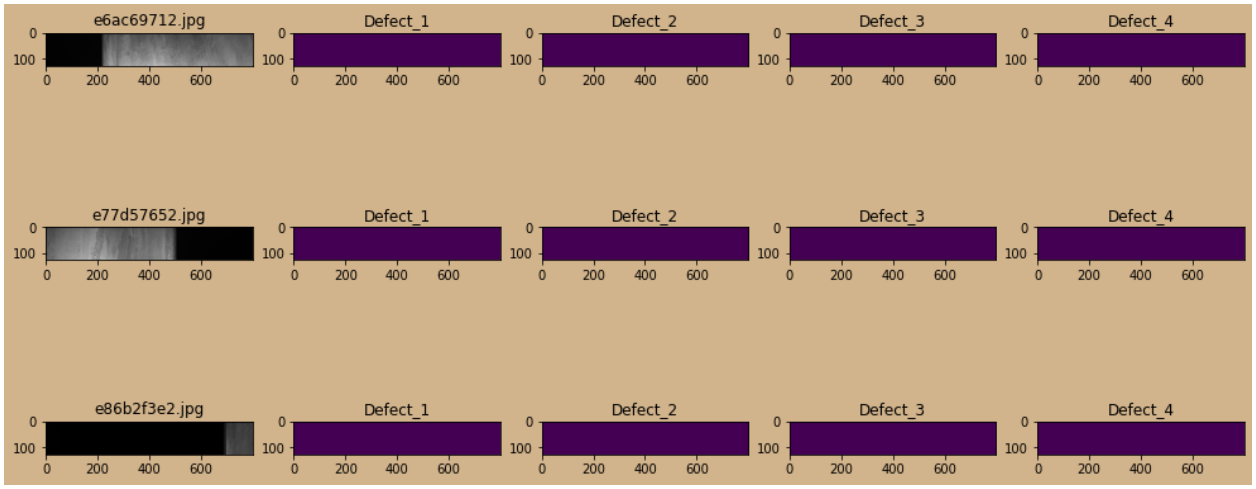
After the training, 10% of the data set which the model hasn't seen yet is passed on to the model for testing. After the model predicts the masks, thresholds on the pixel confidence and the defect area are applied to obtain the actual final masks.

The dice coefficient for the new masks with their actual counterparts = 0.9222982846890861

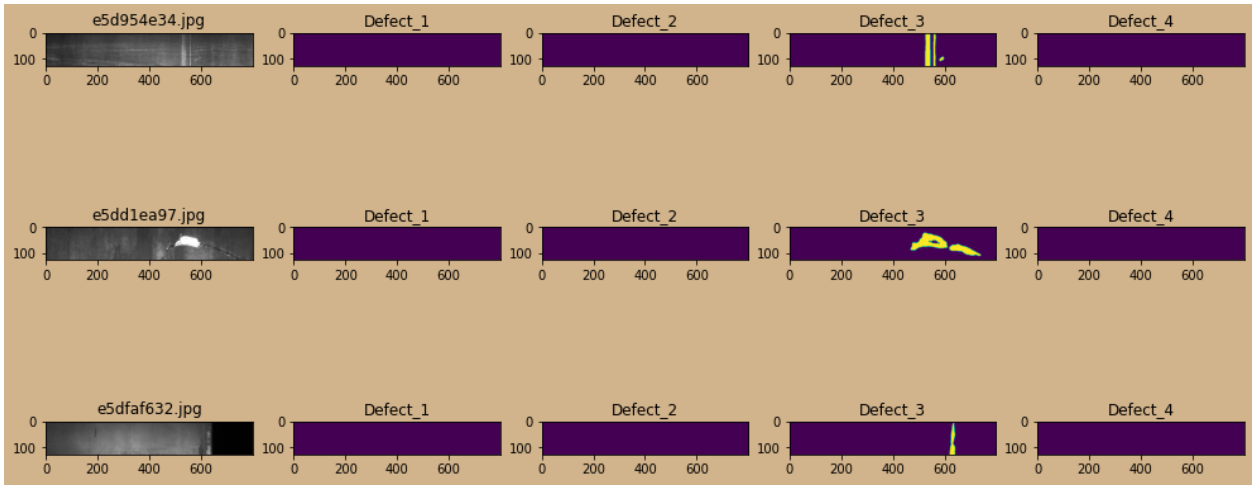
Masks generated for some of the images :
For defect1 :



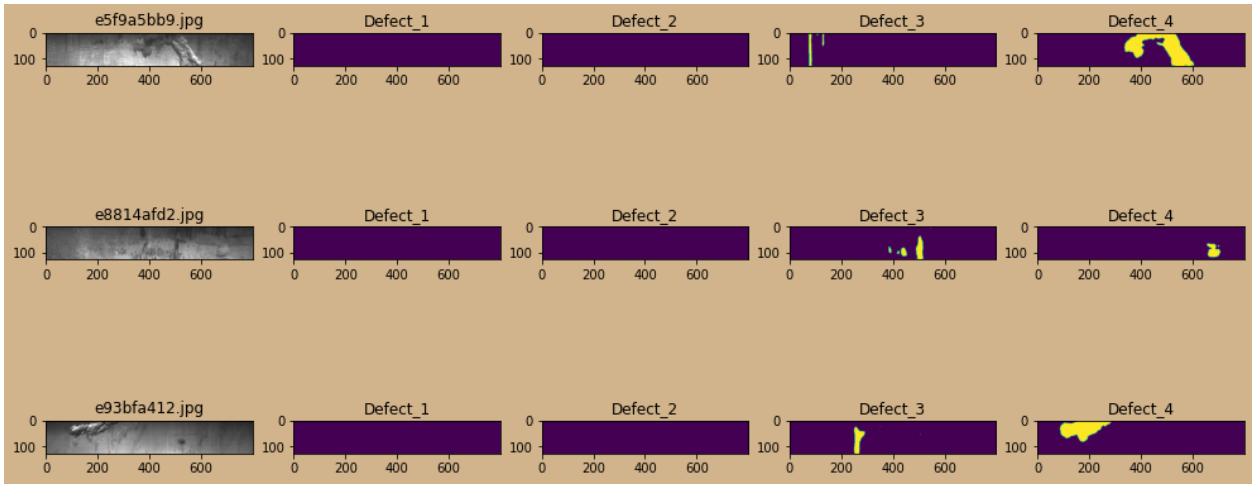
For defect2:



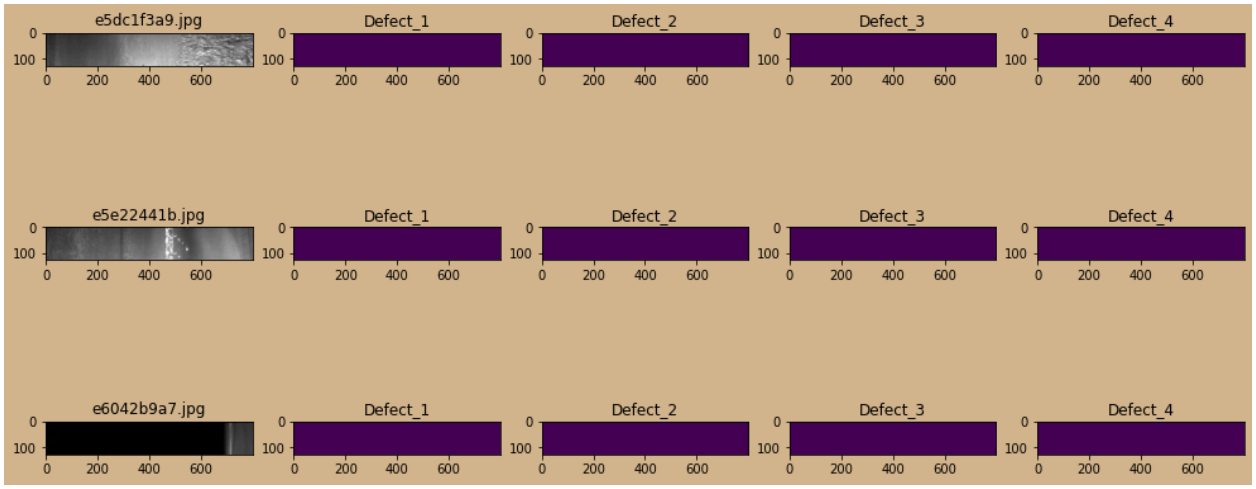
For defect3:



For defect4:



For no_defects :



Conclusions :

The thresholding technique greatly improves the performance of the CNN.

References :

1.Severstal: Steel Defect Detection | Kaggle

2. M. Sharifzadeh, S. Alirezaee, R. Amirfattahi and S. Sadri, "Detection of steel defect using the image processing algorithms," 2008 IEEE International Multitopic Conference, Karachi, 2008, pp. 125-127, doi: 10.1109/INMIC.2008.4777721.

3.https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.711.2081&rep=rep1&type=pdf

4.https://github.com/ml-projects-rana/Steel-Defect-Detection---Image-Segmentation-using-Keras-and-Tensorflow (used for finalising the threshold values)