STEEL DEFECT DETECTION

DETECTING AND CLASSIFYING DEFECTS IN STEEL SHEET IMAGES

Jaysmit Jadhav
jaysmitjadhav@mail.com

ABSTRACT

This projects attempts to solve the problem of steel defect detection by marking the defected area pixel wise in steel sheet images taken by high frequency camera. We explored deep learning methods: Mask R-CNN and U-Net to solve for the task and compared their performance and achieved highest training and validation dice coefficient of 0.8344 and 0.6834 respectively with U-Net.

I INTRODUCTION

Steel is one of the most important building materials of modern times. Steel buildings are resistant to natural and man-made wear which has made the material ubiquitous around the world. The production process of flat sheet steel is especially delicate. From heating and rolling to drying and cutting, several machines touch flat steel by the time it is ready to ship. Before delivering the product, steel sheets need to undergo careful inspection to avoid defects and thus localizing and classifying surface defects on a steel sheet is crucial. Hence, automating the inspection process would accelerate the steel sheet production.

This project is targeted at designing an efficient approach of detecting defects on steel sheets with images from high frequency cameras. We explored different deep learning methods: U-Net and Mask R-CNN to tackle the problem. Our methods get input images of a steel sheet taken by high-frequency cameras and output a same-size segmented image with defected area marked with corresponding defect types.

II RELATED WORK

For steel defect detection, the input is an image of a steel sheet and the output is a same-size segmented image with dense defect area marks. Since the output is a pixel-wise label for the input image, it is essentially a segmentation task. Within image segmentation, there are two different sub-tasks, semantic segmentation and instance segmentation. These problems have been fundamental problems of computer vision for many years and receive great breakthrough since the CNN revival.

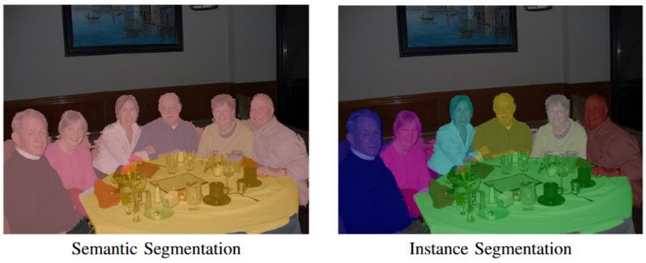


Figure 1 Source: IEEE SIGNAL PROCESSING MAGAZINE, VOL. XX, NO. XX, JANUARY 2018

a. Semantic Segmentation

Semantic segmentation could be considered as a per pixel classification problem. The most popular CNN-based method on semantic segmentation is the Fully Convolutional Network (FCN), which converts fully connected layers into 1x1 convolutional layers and achieves end-to-end per-pixel prediction. However, traditional FCN method suffers from the problem of resolution loss. There are mainly two streams of methods proposed to tackle this problem. The first stream of methods use “atrous convolution”, which enlarges the feature map via linear interpolation. The second utilized deconvolution to learn the up-sampling process.

b. Instance Segmentation

Compared with semantic segmentation, instance segmentation aims to predict not only the class label, but also pixel-wise instance mask to localize varying numbers of instances presented in images. There are mainly two lines of methods to tackle instance segmentation problem: proposal-based methods and segmentation-based methods. Proposal-based methods are closely related to object detection. Mask

R-CNN is the most widely-used method in this stream. The other is segmentation-based, which uses the output of semantic segmentation as input and obtain instance-aware segmentation result later.

III DATASET

The data is provided by Severstal, a company mainly operating in the steel industry. The data contains 6,666 images of steel sheet captured using high frequency cameras. Each image may have a defect of a single class, or defects of multiple classes. The defect region for each image is available. All images are of the size 1600 X 256.

IV EXPLORATORY DATA ANALYSIS

In this section, the data investigated and explored the data to understand the different types of defects in the images, their distribution and visualize the defects on the steel sheet images.

a. Distribution of defects across the classes

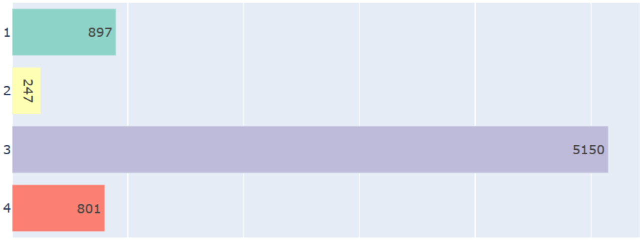


Figure 2 Frequency of images per defect class

There is a large imbalance in the defect classes. 5150 (77.3%) out of the 6666 images have class 3 type of defect. 897 images have class 1, 801 images have class 4 and 247 images have class 2 type of defect.

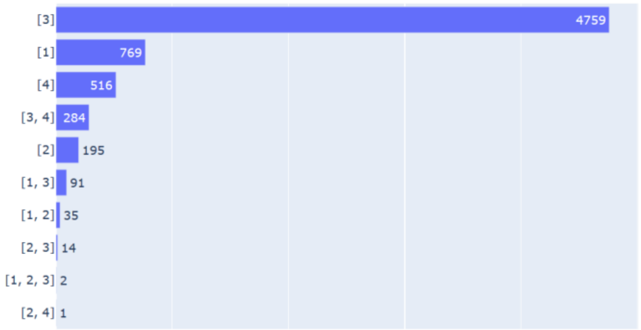


Figure 3 Count of distinct defect combinations in images

Fig.3 shows different defect combinations in the images. 4759 images have only class 3 type of defect. 284 images have defects of class 3 as well as class 4. There are 4 more such combinations. Moreover, there are 2 images that have defects of 3 classes – 1, 2 and 3.

b. Visualizing images and defects

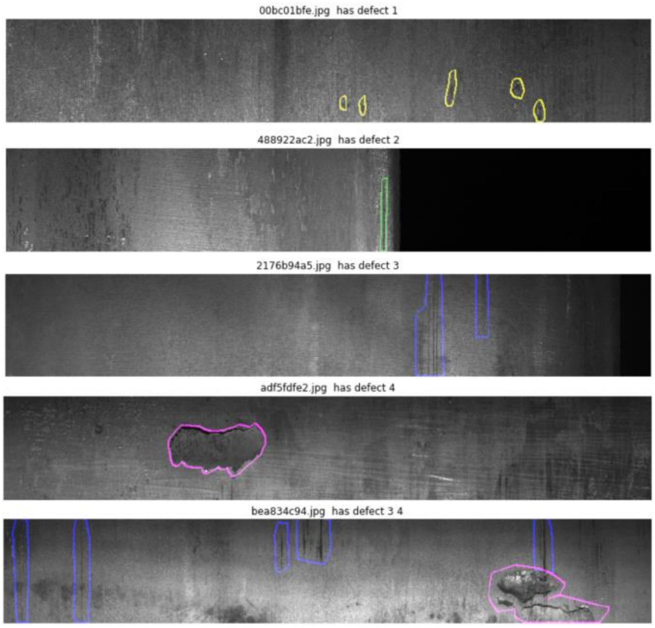

00bc01bfe.jpg has defect 1
488922ac2.jpg has defect 2
2176b94a5.jpg has defect 3
adf5fdfe2.jpg has defect 4
bea834c94.jpg has defect 3 4

Figure 4 Sample images showing defects of different classes with boundaries

V METHODS

a. Mask R-CNN

Mast R-CNN algorithm is an instance segmentation algorithm, which identifies object outlines at the pixel level. It is an extension to Fast R-CNN. It consists of two stages: the first stage scans the image and generates proposals (Region Proposal Network (RPN)) and the second stage classifies the proposals and generates bounding boxes and masks. The final output of Mask R-CNN is stack of masks of the size of input image with bounding box coordinate and defect classes. The architecture is shown in fig. 5.

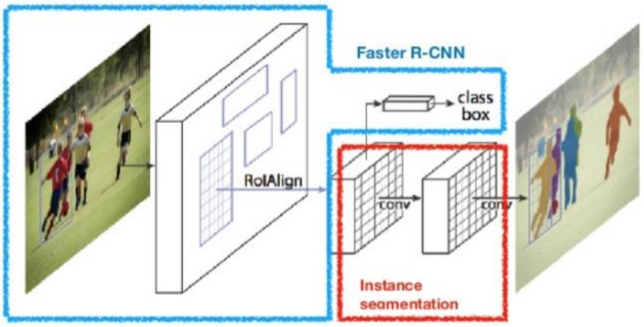


Figure 5 Mask R-CNN architecture

b. U-net

The U-net architecture is shown in fig. 6. It consists of a contracting path (the left side) and an expanding path (the right side). There are 4 down-sampling layers in the contracting path. Each down-sampling layer contains two 3x3 convolutional units, each followed by batch normalization and ReLU, and then a 2x2 max pooling. The contextual information from the contracting path is then transferred to the expanding path by skip connection. There are 4 upsampling layers in the expanding path. Each up-sampling layer contains a 2x2 transposed convolution, a concatenation with the corresponding feature maps from the encoding path, and two 3x3 convolutional units, each followed by batch normalization and ReLU. Finally, there is a 1x1 convolutional layer and a softmax layer to map the feature vector at each pixel to 4 classes.

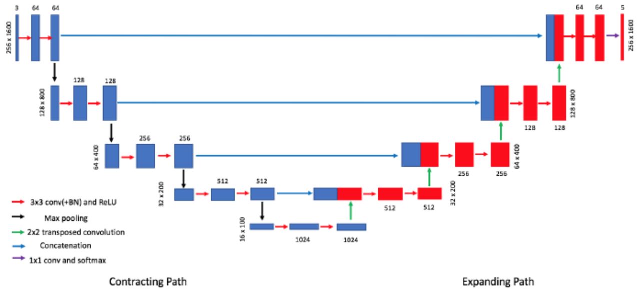


Figure 6 U-net architecture

VI RESULTS

a. 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:

$$Dice(X, Y) = \frac{2.\ |X\ \cap Y|}{|X| + |Y|}$$

Where X is the predicted set of pixels and Y is the ground truth. The Dice coefficient is defined to be 1 when both X and Y are empty.

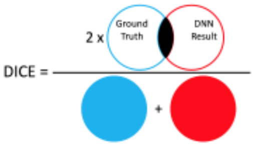


Figure 7 Dice Coefficient

b. Mask R-CNN

The loss function for Mask R-CNN is the combination of the following five terms

1. rpn_class_loss: How well the Region Proposal Network separates background with objects
2. rpn_bbox_loss: How well the RPN localizes objects
3. mrcnn_bbox_loss: How well the Mask RCNN localizes objects
4. mrcnn_class_loss: How well the Mask RCNN recognizes each class of object
5. mrcnn_mask_loss: How well the Mask RCNN segments objects

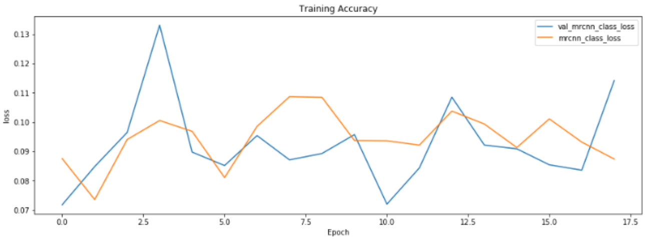


Figure 8 Mask R-CNN training and validation class loss

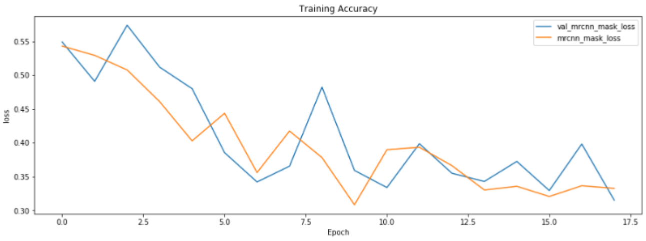


Figure 9 Mask R-CNN training and validation mask loss

Fig 8 shows Mask R-CNN training and validation class loss per epoch. Whereas, Fig 9 shows Mask RCNN training and validation mask loss. Throughout the training process, the rpn_bbox_loss keeps at a high value and accounts for the majority of the loss. The result shows that the Mask R-CNN network is not an ideal model for the defect detection task.

The instance segmentation algorithm seems not to perform in this defect detection task due to several reasons. On one hand, Mask R-CNN is known for overfitting easily. In the dataset, some labels are too rare to train a good Mask R-CNN network even with data augmentation, which gives little boost. On the other hand, the dataset does not match the assumptions of instance segmentation very well. The defects in reality do not have clear boundaries and typical shapes. The various shapes and unclear boundaries cause the region proposal network to fail to propose good enough candidate regions. That explains why the majority of the loss come from the region proposal network.

c. U-net

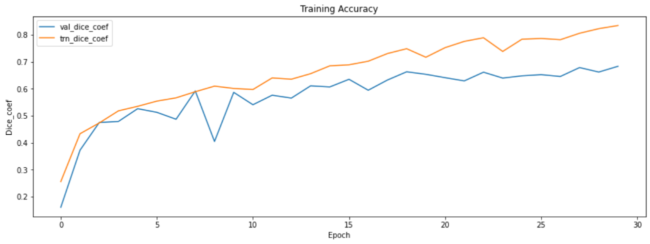
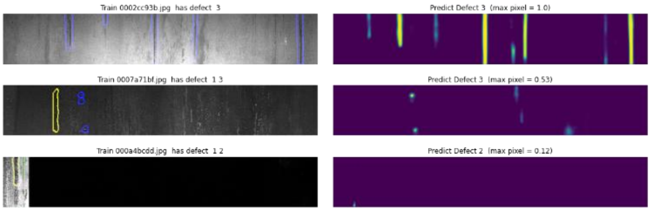


Figure 10 U-net training and validation dice coefficient

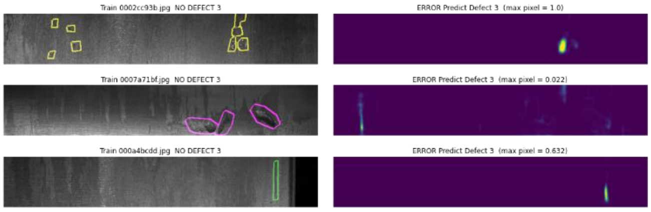
After 30 epochs, the training dice coefficient is 0.8344 and the validation dice coefficient is 0.6834. Fig 11 shows one prediction mask for a defect that is present in the images.



Train 0002cc93b.jpg has defect 3
Predict Defect 3 (max pixel = 1.0)
Train 0007a71bf.jpg has defect 1 3
Predict Defect 3 (max pixel = 0.53)
Train 000a4bcdd.jpg has defect 1 2
Predict Defect 2 (max pixel = 0.12)

Figure 11 Images with maximum pixel value in the titles, measure of mask confidence

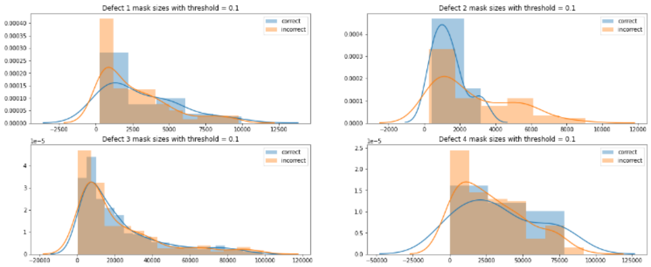
The dataset is imbalanced with 77.3% of the images being as type 3 defect. Fig 12 shows images with defect type 1, 2 and 4 along with predicted defect mask of type 3 defect.



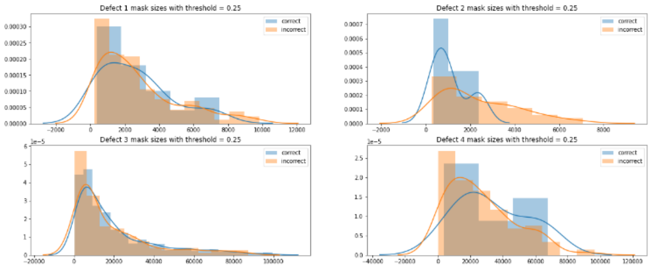
Train 0002cc93b.jpg NO DEFECT 3
ERROR Predict Defect 3 (max pixel = 1.0)
Train 0007a71bf.jpg NO DEFECT 3
ERROR Predict Defect 3 (max pixel = 0.022)
Train 000a4bcdd.jpg NO DEFECT 3
ERROR Predict Defect 3 (max pixel = 0.632)

Figure 12 Images with defect types 1, 2 and 4 and mask type 3

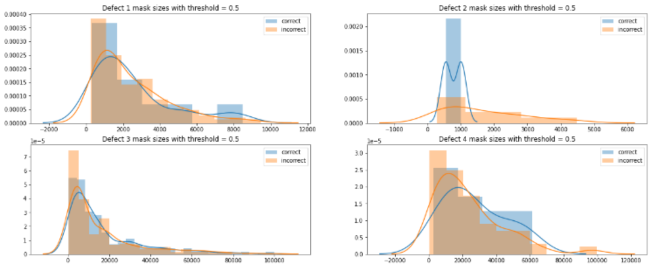
Fig 13 compares the distribution of incorrect versus correct masks for different thresholds for each defect type i.e. the predicted size of each defect mask. We would hope that if an image does not have a particular defect then UNET would not predict a mask (i.e. predict less than 250 pixel mask). This is not the case. When UNET predicts a mask when a defect isn't present, we call that an incorrect mask. When UNET predicts a mask when a defect is present, we call that a correct mask. If UNET predicts less than 250 pixels, we will treat that as no mask predicted.



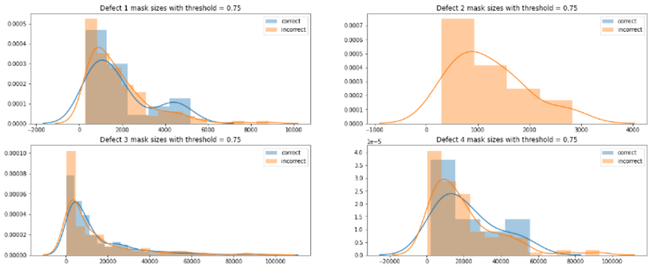
a Threshold 0.1



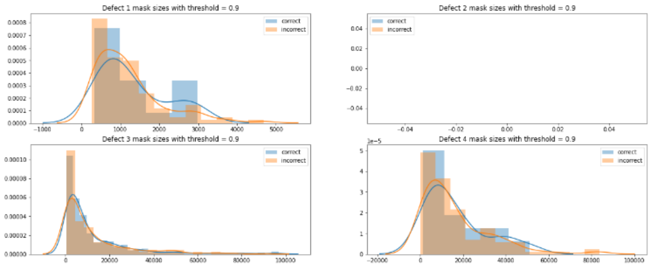
b Threshold 0.25



c Threshold 0.5



d Threshold 0.75



e Threshold 0.9

Figure 13 Histograms of predicted size of each defect mask for various thresholds

VIII CONCLUSION

This project leverages the segmentation and instance segmentation computer vision techniques to solve the steel defect detection problem. The models outputs the pixel wise defected area with corresponding labels. Among all the models, the U-Net performs the best with the training dice coefficient as 0.8344 and the validation dice coefficient as 0.6834.

IX FUTURE WORK

Due to the imbalance between different defect labels, more data augmentation processes such as rotation and scaling on the images should be performed on the rare defect classes. Additionally, effort could be put to predict the severity of each defect kind so that the steel factory could also leverage the severity information to determine whether to dispose the steel sheet.

ACKNOWLEDGEMENT

Thank you to Dr. Kevin Ding for all the continuous support with this project.

CODE: kaggle.com/jaysmit/u-net