Steel defect detection with high-frequency camera images
FA 19-20 CS 229 Project

Mengxi Li, Xinrui Wang, & Zhuoran Ma
School of Engineering
Stanford University
Stanford, CA 94305, USA
{mengxili,xinrui,zma2}@stanford.edu

Abstract

We focus on the problem of steel defect detection, where we are given images of a steel sheet taken by high-frequency cameras and aim to mark pixel wise defected area. We explored three deep learning methods: Xception, U-Net and Mask R-CNN to solve for the task and compared their performance and achieved highest Dice coefficient of 0.492 with U-Net.

1. Introduction

Steel is one of the most important building materials of modern times. And 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's 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 a more efficient approach of detecting defects on steel sheets with images from high frequency cameras. We explored three different deep learning methods: Xception [3], U-Net [12] and Mask R-CNN [6] 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. We concluded that among the three methods we tried, U-Net achieves highest performance with a Dice coefficient of 0.492.

2. 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's essentially a segmentation task. Within image segmentation, there are two different sub-task, 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.

2.1. Semantic Segmentation

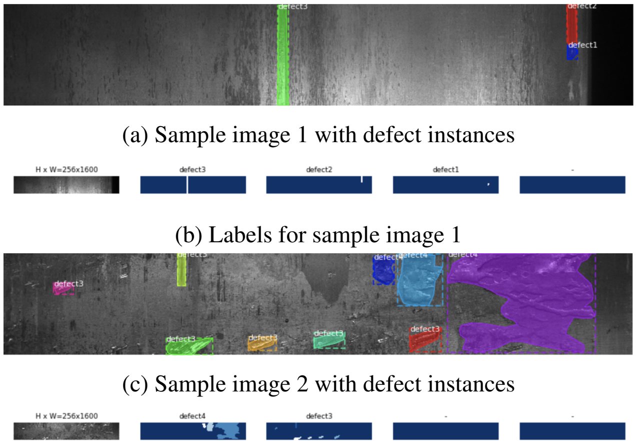
Semantic segmentation could be considered as a per-pixel classification problem. The most popular CNN-based method on semantic segmentation is the Fully Convolutional Network (FCN) [9], which converts fully connected layers into 1x1 convolutional layers and achieves end-to-end per-pixel prediction. However, traditional FCN method suffers from the problem of resolution loss. There are mainly two main streams of methods proposed to tackle this problem. The first stream of methods [2] use "atrous convolution", which enlarges the feature map via linear interpolation. The second utilized deconvolution [10] to learn the upsampling process.

2.2. Instance Segmentation

Compared with semantic segmentation, instance segmentation aims to predict not only 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 [5, 7]. Mask R-CNN is the most widely-used method in this steam, which proposes to add a fully convolutional network branch based on [7] and achieves great performance on [8, 4]. The other is segmentation-based, which uses the output of semantic segmentation as input and obtain instance-aware segmentation result later. Among them, [15, 14] proposed to use a graphical model to infer the order of instances and [11, 13] utilized RNN to obtain one instance in each time step.

3. Dataset and Features

The dataset we are using is one consisting of high-frequency images of steel sheet, corresponding defects classes, and defect regions. This dataset is obtained from [1]. The images provided are of size 1600 × 256 × 1 and totals 12568 counts. Out of all the images, 5902 images are are with defects and 6666 images are without defects. There are four labels of defects 1, 2, 3, and 4. In the images with defects, 897 images are of class 1 defect, 247 images are of class 2 defect, 5150 images are of class 3 defects, and 801 images are of class 4 defect. Additionally, 6239 images have only 1 class of defect, 425 images have two classes of defects, and only 2 images have 3 classes of defects. Two samples are shown in Fig. 1



(a) Sample image 1 with defect instances

(b) Labels for sample image 1

(c) Sample image 2 with defect instances

(d) Labels for sample image 1

Figure 1: Sample image with instance-level defects

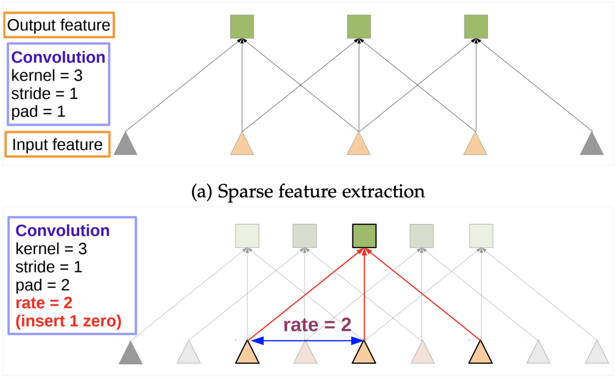
4. Methods

4.1. Xception

We utilize Xception [3] as our baseline method. We removed the last fully-connected layers of the original Xception model and followed the idea of "atrous convolution" as used in [2], which enlarges the feature map by linear interpolation. The "atrous convolution" used in our method is illustrated in Fig. 2.

4.2. 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 structure is shown in Fig. 3.



(a) Sparse feature extraction

(b) Dense feature extraction

Figure 2: Illustration of atrous convolution in 1-D. (a) Sparse feature extraction with standard convolution on a low resolution input feature map. (b) Dense feature extraction with atrous convolution with rate r = 2, applied on a high resolution input feature map.

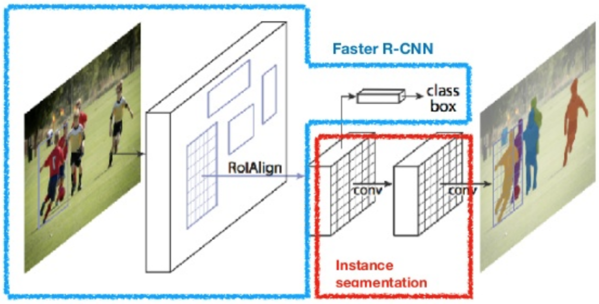


Figure 3: Mask R-CNN Structure

4.3. U-net

The U-net architecture is shown in Fig. 4. 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 up-sampling 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 an 1x1 convolutional layer and a softmax layer to map the feature vector at each pixel to 5 classes (0 for no defect, 1-4 for defects of different classes).

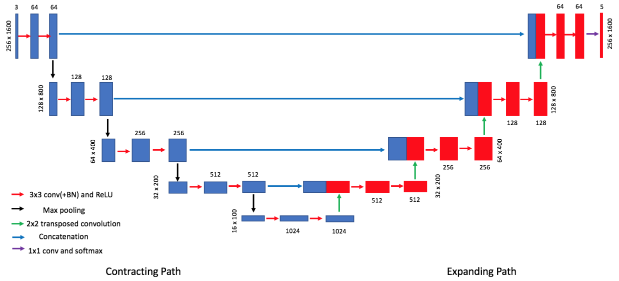


Figure 4: U-net Architecture

5. Experiments & Results

5.1. 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:

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

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.

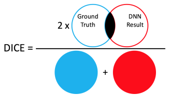


Figure 5: Dice Coefficient

5.2. Xception

The loss function we adopted during training is defined as

$$L = L_{dice} + \alpha * L_{CE} \tag{1}$$

where $L_{dice} = \frac{2*TP}{2*TP+FP+FN}$ $(TP, FP, FN$ represent True Positive, False Positive and False Negative respectively), L_{CE} is the traditional cross-entropy loss. α is set to 1 in the training process. The images are scaled to $256x256$ during training. We train our Xception-based model for 6 epochs. Learning rate during the training process is set to $1e-3$. We use adam as the optimizer. The binary cross-entropy loss, dice score and IoU score during the training are plotted in Fig 6.

In the inference phase, images are first resized to $256 * 256$ to feed as the input of our model. The output of the model are then resized back to $256 * 1600$ as the final predictions. Dice score is used as our evaluation metric. Our model achieves 0.369 on positive dice score and 0.978 on negative dice score. We also show a qualitative result of our model in Fig 7.

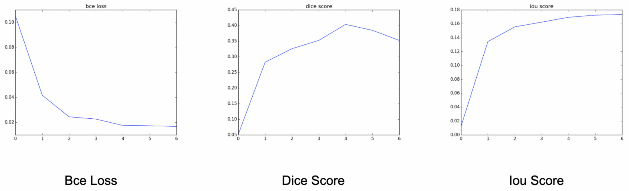


Figure 6: Binary cross-entropy loss, dice score and IoU score during the training process of Xception model.

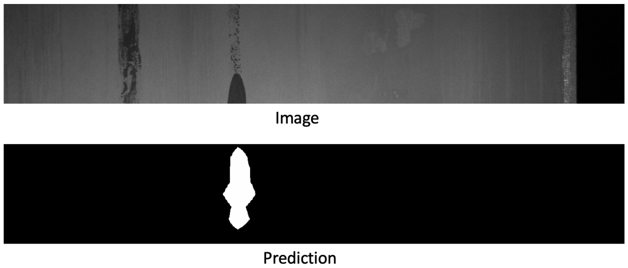


Figure 7: Qualitative result of Xception model.

5.3. 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 objetcs

2. rpn_bbox_loss : How well the RPN localize objects

3. mrcnn_bbox_loss : How well the Mask RCNN localize objects

4. mrcnn_class_loss : How well the Mask RCNN recognize each class of object

5. mrcnn_mask_loss : How well the Mask RCNN segment objects

The training loss and validation loss is shown in the Fig. 8a. The training loss quickly drops to around 1.1 and remain steady throughout the training process. The average positive Dice score is 0.083 and the negative Dice score is 0.898. The model is not able to provide accurate shape and location of the defect as seen in the labeled image, which is shown in Fig. 8b and 8c. 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.

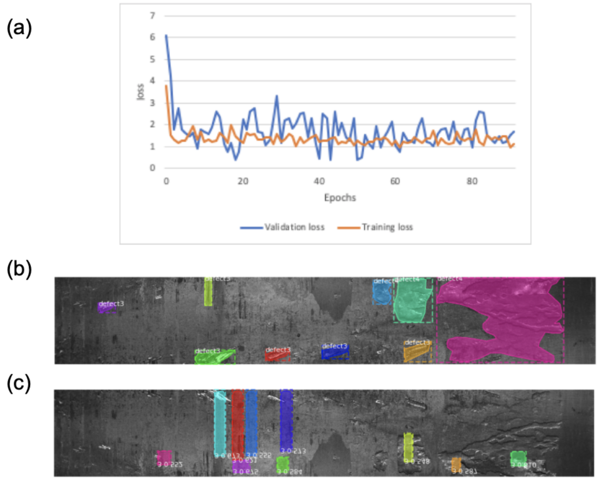


Figure 8: Results of the Mask R-CNN. (a) Loss of the training and the validation sets. (b) A sample labeled image containing instances of defect. (c) The defect instances prediction result generated by Mask R-CNN

5.4. U-Net

We used Dice loss function during training, which is defined as

$$L_{dice} = 1 - \frac{1}{B}\sum_{i=1}^{B}\frac{2\sum_{j=1}^{n} y_j^{(i)}\hat{y}_j^{(i)}}{\sum_{j=1}^{n} y_j^{(i)} + \sum_{j=1}^{n}\hat{y}_j^{(i)}} \qquad (2)$$

where B is the batch size, $n = 4 \times 256 \times 1600$ is the total number of pixels of the 4 defect classes, $y_j^{(i)}$ is the true label of each pixel (1 for defect and 0 for normal), and $\hat{y}_j^{(i)}$ is the predicted probability. A batch size of 4 was used due to the limitation of GPU memory. We used adam as the optimizer. The loss values and the positive Dice score during training are shown in Fig. 9a and 9b, respectively. After 20 epochs, the positive and negative Dice scores are 0.492 and 0.977. The U-net model showed good performance for predicting defects of class 3 (Fig. 9c). However, it failed to predict many defects of class 1, 2 and 4 (Fig. 9d).

6. Discussion

6.1. Xception

During the evaluation process, we found that our model suffers from the data insufficiency problem. The prediction of our Xception model could mainly capture defect type 3. That is because the majority of the defect data in our training set belongs to type 3. We also found the prediction of our model is quite coarse. It is because we directly upsample the feature map 32 times, thus losing some details in the training process.

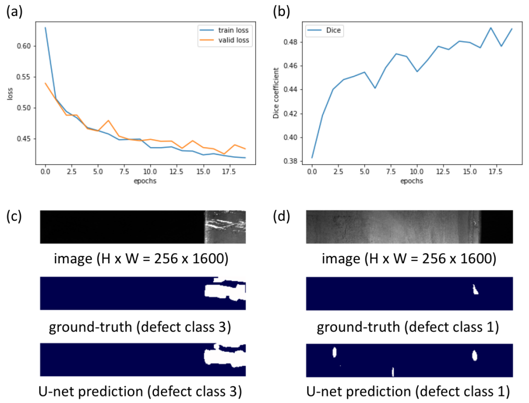


Figure 9: Results of the U-net. (a) Loss of the training and the validation sets. (b) Average positive Dice score of the validation set during training. (c) A sample image containing defect of class 3. (d) A sample image containing defect of class 1.

6.2. Mask R-CNN

The instance segmentation algorithm seems not to perform in this defect detection task due to several reasons. On the 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.

6.3. U-Net

The loss value of the validation set is similar to that of the training set, which indicates that the model generalizes well to other sets. However, due to unbalanced classes, the U-Net model can only predict defects of class 3, which is the majority class of the data set.

7. Future Work

Due to the imbalance between different kinds of labels, more data augmentation processes such as rotation and scaling on the images should be performed on the rare defect classes. Other than that, we could train an embedding specifically for steel high-frequency images for feature extraction such that we could train a logistic regression quickly whenever a new defect a feature is of interest. Lastly, 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.

8. 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 a positive dice coefficient of 0.492.

9. Contributions

Mengxi is responsible for the training and benchmarking the Xception model. Xinrui is in charge of training and benchmarking the Mask R-CNN network. Zhuoran is in charge of training and benchmarking the U-Net model. The code for this project can be found here: https://drive.google.com/open?id=1Uk-fuColKcOjBCFX3tG2W4jSa7fTz8G6

References

[1] Severstal: Steel defect detection. 2

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1, 2

[3] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1, 2

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 1

[5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[7] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[9] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[10] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015. 1

[11] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016. 1

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[13] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European conference on computer vision*, pages 467–482. Springer, 2016. 1

[14] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 669–677, 2016. 1

[15] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2614–2622, 2015. 1