Steel defect detection with deep semantic segmentation

Ching Pui WAN
HKUST
cpwan@ust.hk

Cheuk Wah NG
HKUST
cwngaj@connect.ust.hk

Chung Hon TSE
HKUST
hctseaa@ust.hk

Abstract

This work focuses on the dense prediction of steel defect by deep semantic segmentation. The dataset consists of 4 classes of defects and was taken from Kaggle. The performance of two models, U-net and Context Aggregated Network (CAN), were evaluated in this work. Both models are convolutional-based and trained against the binary cross entropy loss. Under the current setup, CAN performs significantly better than U-net. However, the baseline model which is observed from Kaggle, still performs slightly better than CAN. Dice score was used to evaluate the performance. Failed cases were studied and some enhancements were proposed to address the incapability of the models. The code used in the project can be found here.

1. Introduction

Steel production is one of the most important industrial processes. The production process of flat sheet steel is especially delicate. This contributes to the demand for the automation of the defects detection of the flat sheet steel. Images from high-frequency cameras are used for defect detection. Localizing and classifying surface defects on a steel sheet is a crucial part of the automation of defect detection. With accuracy and efficient defect detection algorithms, the manufacturing standards can be maintained. This project is inspired by the 'Severstal: Steel Defect Detection' [1] competition on Kaggle.

1.1. Formulation

We formulate the defect detection problem as a segmentation problem: given a high-frequency image of the surface of a steel sheet, classify any kind of defects presented in each pixel. We will attack the problem in two ways, the supervised way and the semi-supervised way. In the supervised approach, we will use all the labeled data. In the semi-supervised approach, we make use of a small subset of the data and explore the generality of our algorithms, such as generalizing to more kinds of defects.

1.2. Related work

For the semantic segmentation problem, two approaches have been studied: the supervised approach and the semi-supervised approach. For the supervised approach, recent works include U-Net [2] and Context Aggregation Network [3, 4]. U-Net is neural network with symmetric structure across the bottleneck layer, for which skip connections are added across the symmetric layers. U-Net allows different level of features to be encoded in the later layer. Context Aggregation Network uses dilated convolutional layers and gives a larger receptive field in later layer to facilitate the learning from global features. For the semi-supervised approach, recent works on few-shot learning [5, 6] and meta-learning [7] provides an alternative way to transfer better a pretrianed model for a wider range of applications. The works suggest a way to deal with minority classes in semantic segmentation problem.

2. Problem statement

2.1. Datasets

The dataset provided by Serverstal on Kaggle [1] will be used for training and testing. It consists of 12568 steel images for the training set, annotated with 4 kinds of possible defects. Despite the availability of the test set, the ground truth is not publicly revealed. Therefore, we will further split the original training set to a custom training set and a custom test set at the ratio of 90/10. The proportion of 4 kinds of defects within a set will be maintained during the split.

2.2. Baselines

We will evaluate our results on both accuracy and efficiency. For metric on the accuracy, we will use the Dice coefficient:

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

for X being the predicted sets of pixels and Y being the ground truth. The Dice coefficient measures how similar the predicted sets and the ground truth is. The Dice coefficient is 1.0 for the perfect alignment of predictions and ground truth. As of Oct 10, 5 pm, the best score in public leaderboard in Kaggle is 0.92209. That of the rank 50th (out of > 2500 submits) is 0.91336. We will take 0.91336 as our benchmark.

For evaluating efficiency, we aim to restrict the influence time allowed for our custom test set to be within an hour, which corresponds to less than 3 seconds per image. As a bonus if time allows, we may take a challenge to train a few shot learning model in one hour.

3. Technical Approaches

Few existing implementations of semantic segmentation like U-net[2] and Context Aggregation Network[3] will be tried to attack the steel defects detection problem with the baseline performance. Attempts will be made to change the network pipelines and derive a more optimal loss function in order to achieve a better performance in terms of accuracy.

Furthermore, to explore the generality of the defect detection algorithm, transfer learning approach and few shot learning approach will be evaluated. One way is to pre-train a model with datasets for similar tasks followed by fine-tuning it with a limited number of training data in our dataset. Another approach is meta-learning in which an extra network is trained to predict hyperparameters and pipeline compositions. By doing so, the learning on steel defect detection task will become more efficient and therefore require less number of training data.

3.1. U-Net

U-Net[2] is a fully convolutional network (FCN) which learns segmentation in end to end setting from a raw image to the segmentation map.

The architecture of U-net is depicted in Figure 1. U-Net architecture is separated in 3 parts. They are the contracting/down-sampling path, bottleneck and the expansive/up-sampling path. U-net is symmetric and its skip connections between the down-sampling path and the up-sampling path apply a concatenation operator instead of a sum. These skip connections intend to provide local information to the global information while up-sampling. Because of its symmetry, the network has a large number of feature maps in the up-sampling path, which allows to transfer information. A modified U-net instead of the canonical one [2] is used in this work. In contracting/down-sampling part, the state-of-the-art ResNet101 is used for retrieving features. It contains 5 blocks. The first block contains a 7x7 convolutional layer with stride 2. Each of the following 4 blocks contain 2 3x3 convolutional layers with stride 2. The input and output of these 4 blocks are short-circuited to predict the residual value instead of the original signal. The next part is the bottleneck which is an average pooling layer which output a 1000-dimension feature. The feature is

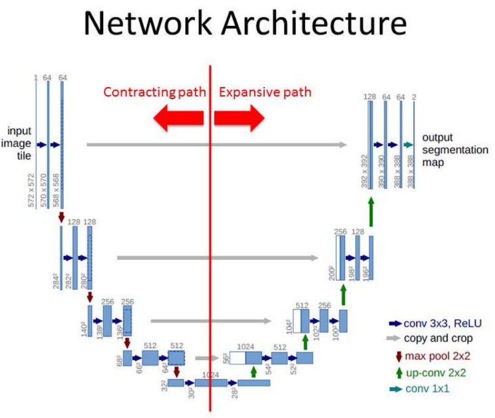


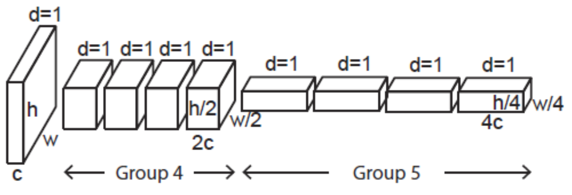
Figure 1: The symmetric structure of U-Net across the bottleneck. Skip connections connect the expansive/up-sampling path and the contracting/down-sampling path.

fed to a softmax function. Finally, there is an expansive/up-sampling path which is composed of 4 blocks. Each block is composed of a 3x3 convolutional layer with stride 1 followed by an up-sampling convolutional layer. The input of each block in the up-sampling path is the output of the last layer concatenated with the corresponding cropped feature map from the contracting path. Unlike the canonical U-net, the number of output channels are changed to the number of classes in our task, which is 4, in order to predict the probability of each defect class for a given pixel. The training was started with a pre-trained model for ImageNet [8].

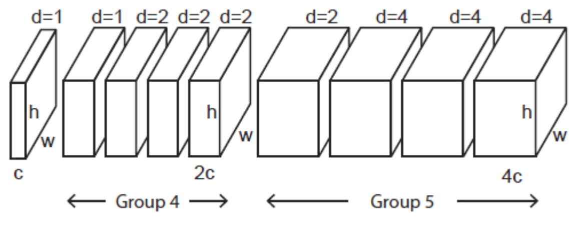
3.2. Context Aggregated Network

Context Aggregated Network[3], also called Dilated Residual Networks (DRN) [4] , aims to preserve spatial resolution in convolutional networks for image classification problems.

It converts from ResNet to make a DRN. The changes are depicted in Figure 2. It has the same design for the first 3 groups of layers in ResNet. However, start from group 4, it removed the striding in group 4 and group 5, and replace it with 2 dilated and 4-dilated convolutions layer respectively. In ResNet the output of these two layers will from 28x28 drop to 14x14 and 7x7, but in DRN, it will not drop, the group 5 output will also be 28x28 drop to 14x14 and 7x7, but in DRN, it will not drop, the group 5 output will also be 28x28. DRN is designed for image classification as it can output high-resolution images map and makes use of removing max pooling and adding layers which performs better result than ResNet. In particular, we employed the DRN-D-22 model described in the source code of the paper [4].



(a) ResNet



(b) DRN

Figure 2: Comparison of Dilated Residual Network (DRN) and ResNet. In DRN, the spatial resolution is preserved across layers. Here shown an example of converting the last 2 groups of VGG into DRN structure.

3.3. Binary cross entropy loss

Binary cross entropy loss[9] (a.k.a Sigmoid Cross-Entropy loss) is a sigmoid activation followed by a cross-entropy loss which is commonly used for multi-label classification. The insight of the loss is that an element belonging to a certain class should not influence the decision for another class. The “binary” in the name of the loss is due to that the loss sets up a binary classification problem for “Positive” and “Negative” classes for each class label. So when using this loss, the formulation of Cross Entropy Loss for binary problems is often applied:

$$\begin{aligned} CE &= -\sum_{i=1}^{2} t_i log(f(s_i)) \\ &= -t_1 log(f(s_1)) - (1-t_1) log(1-f(s_1)) \end{aligned}$$

for which s_i is the predicted probability of the class label (“Positive” or “Negative”) and t_i is the ground truth label.

3.4. Dice loss

Dice loss is commonly use in image segmentation or bounding box problems. For a single class, the dice loss DL $(p, \hat{p}) = 1 - \frac{2p\hat{p}+1}{p+\hat{p}+1}$ for the ground truth label $p \in \{0, 1\}$ and the prediction $\hat{p} \in [0, 1]$. The dice loss for multiple classes is the arithmetic mean of the dice loss for each classes.[10] It is believed that the dice loss can account for the global information for the segmentation problem since it seems to balance the true positive and false positive for each classes by the averaging of dice loss with respect to each classes.

3.5. Indeterminacy loss

It is discovered that the network is reluctant to predict defect with high confidence given the aforementioned dice loss. For a particular defect class of a sample point, when both prediction and ground truth are empty sets, the dice loss are defined as 0. Since the occurrences of defect class 1, 2 and 4 in the dataset are not as frequent as the defect class 3, the network can achieve a low loss by simply predicting no defect in most cases. To alleviate that, a constant Indeterminacy loss is introduced to regulate such indeterminacy. It is defined as

$$DL = -log(1/n)$$

for which n is the number of defect classes, 4. The intuition is that an indeterminate prediction is as bad as randomly picking a defect class out of 4. The indeterminacy loss is used jointly with the dice loss such that

$$L = \begin{cases} \text{DL}, if \exists i \in \{1, 2, 3, 4\}, \text{such that} f(s_i) >= 0.5 \\ \text{IL}, otherwise \end{cases}$$

4. Results

4.1. Setup

The dataset of 12568 images was split at a ratio of 90/10. The training set consist of 90% of the dataset while only 6000 of them are images of steel with defects. In the training stage, we fed our models with these 6000 images. We fed the whole image into our models, with a $\frac{1}{2}$ chance of horizontal flipping. Our models were trained agaist the BCE Loss.

In the testing stage, we inferenced our models on the 10% of the dataset. Note that there were 4 channels in the output layer of our models, each entry represented the probability of a pixel being the particular kind of defects. At each pixel, we classify the defect type to be the one with the maximum value among the 4 channels. Also, we used a threshold of 0.5, which means, if the maximum among the 4 channels was not greater than 0.5, the pixel would have predicted as no defect. In other words, the prediction at pixel $[i, j]$ from the channel output C_1, C_2, C_3, C_4 is :

$$\text{pred}[i, j] = \begin{cases} \text{no defect}, if \max_n C_n[i, j] < 0.5 \\ \text{defect}[\arg\max_n C_n[i, j]], otherwise \end{cases}$$

With the prediction described in this subsection, we evaluated the dice score for the 1257 images in the test set at each epoch.

4.2. U-net

The training took 8 hours in total for 100 epochs. The model achieved a dice score 0.6025 after training for 4 epochs. The best dice score **0.7851** was achieved at epoch 71.

4.3. Context Aggregated Network

The training took 20 hours in total for 80 epochs, which achieved a dice score of **0.8891**. Notably, the model achieved a dice score of 0.8092 after training for 4 epochs, which took an hour.

4.4. Ablation study

4.4.1 Combinations of loss for CAN

We perform ablation study for the CAN model on different combination of loss function. We use the CAN trained alone with BCE Loss as a pre-trained model. Then we trained an additional 5 epochs with a combination of loss, to investigate the impact of using a different loss function. In both cases, we gave an equal weight to the BCE Loss and the other loss. The combinations of loss we investigate are:

1. BCE Loss+ Cross Entropy
2. BCE Loss+ Dice Loss

The results are summarized in Figure 3 and Table 1. Incorporating a Cross Entropy Loss degrades the model in terms of both Dice Score and IoU. For the combination of BCE Loss +Dice Loss, the Dice Score is boosted as illustrated in Figure 3. However, when inspecting the Intersection over Union score (IoU) for each type of defects, adding Dice Loss in fact degrades the model.

This contradiction is due to the bias of the Dice Score:

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

When both the ground truth label Y and the predicted label X occupies no pixels. the Dice Score gives a score of 1 in limit. This may encourage the model to make a conservative choice to predict a pixel to be no defect. When checking the empirical results, the model trained with Dice Loss indeed predicts no defect for most of the pixels. Therefore, we can say that the model has cheated by not predicting any defects.

Combination of Loss	Type 1	Type 2	Type 3	Type 4
BCE Loss	62.94	17.35	94.62	90.77
BCE Loss+ Cross Entropy	4.97	0.00	59.43	53.99
BCE Loss+ Dice Loss	5.53	0.00	81.34	0.05

Table 1: The Intersection of Union for each type of defects in the test set under different combinations of loss used in the training of CAN.

4.4.2 Indeterminacy loss for U-net

The trained U-net produces indeterminate results to get a high accuracy. After re-training the network with the introduction of indeterminacy loss. The network no longer

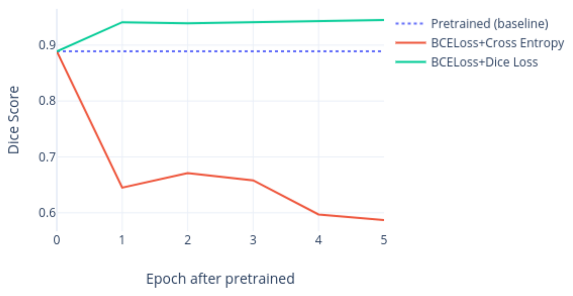


Figure 3: The performance of CAN using different combinations of loss. Using a pre-trained CAN model, the model is further trained for 5 epochs with different combinations of loss. Adding Dice Loss seems to boost the performance.

produces empty predictions on sample points with a non-empty ground truth for the high accuracy. Table 2 shows the IoU of models trained with

1. BCE Loss
2. BCE Loss + Indeterminacy Loss

After switching the feature extraction layers from ResNet34 to ResNet101, the network finds a way to exploit the loss function by predicting indeterminate results. Therefore, the IoU becomes all 0. With the introduction of Indeterminacy Loss, the IoU for the defect type 3 has boosted to 81.37. However, the optimal constant penalty is yet to be found in order to give more balanced predictions among the defect classes.

Combination of Loss	Type 1	Type 2	Type 3	Type 4
BCE Loss (with ResNet34)	9.57	2.07	83.67	79.48
BCE Loss (with ResNet101)	0.00	0.00	0.00	0.00
BCE Loss+ I-Loss	0.00	0.00	81.37	00.00

Table 2: The Intersection of Union for each type of defects in the test set under different combinations of loss used in the training of U-net.

4.5. Failed cases

A major weakness of our models are that

1.) False Positive could not be accounted. Since we trained our models only with images that contain defects. The models falsely recognize the designed patterns on the steel as some types of defects. An example is given in Figure 4c.

2.) Class imbalance could not be accounted. The Intersection of Union (IoU) for type 1 and type 2 defects for

both our models (CAN and U-net) were lower than the other defect by at least a 30%. This might due to that both types of defects occupies a smaller region compare to other type of defects. An example is given in Figure 4d.

5. Conclusion

We have performed experiments on U-net and CAN on the segmentation problem for steel defect detection. We achieved a best Dice Score of 0.7851 and 0.8891 for U-net and CAN respectively. We performed ablation study on loss function, including a mixture of Binary Cross Entropy Loss, Dice Loss, and Cross Entropy Loss. Yet we found that using alone the Binary Cross Entropy Loss gave the best performance. We found the problem of Dice Loss being bias to conservative prediction. We experimented with a novel indeterminacy loss for punishing the abuse of Dice Loss.

For the future development, we suggest a wider exploration of segmentation model, such as Fully Convolutional Networks[11] or Google Deeplab[12, 13]. We suggest an ensemble of segmentation models trained independently on minority classes with the main segmentation models we describe in this project. We also hope our segmentation results can encourage further research on similar context.

References

[1] Severstal, "Severstal: Steel defect detection can you detect and classify defects in steel?." =https://www.kaggle.com/c/severstal-steel-defect-detection/data, 10 2019.

[2] 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*, pp. 234–241, Springer, 2015.

[3] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[4] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[6] I. Ustyuzhaninov, C. Michaelis, W. Brendel, and M. Bethge, "One-shot texture segmentation," *arXiv preprint arXiv:1807.02654*, 2018.

[7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, JMLR. org, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

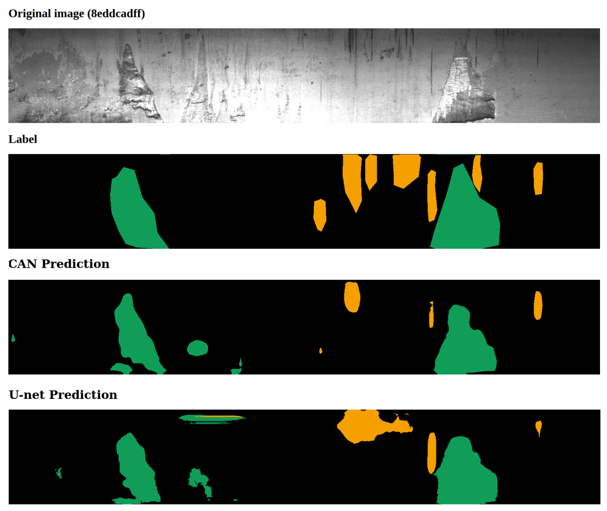
[9] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[10] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pp. 240–248, Springer, 2017.

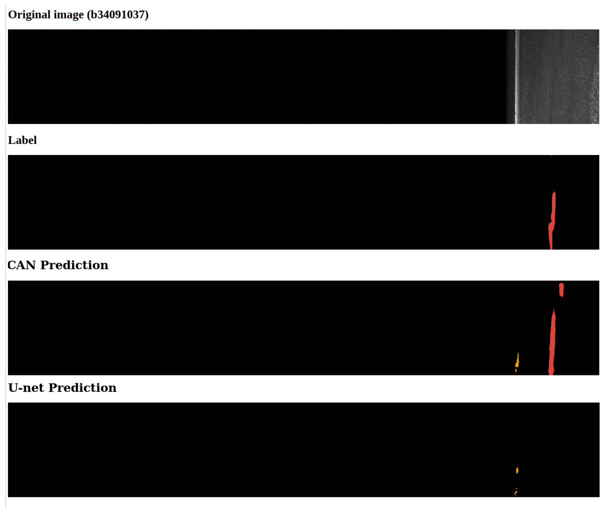
[11] T. D. Jonathan Long, Evan Shelhamer, "Fully convolutional networks for semantic segmentation," *arXiv preprint arXiv:1411.4038v2*, 2015.

[12] I. K. K. M. A. L. Y. Liang-Chieh Chen, George Papandreou, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

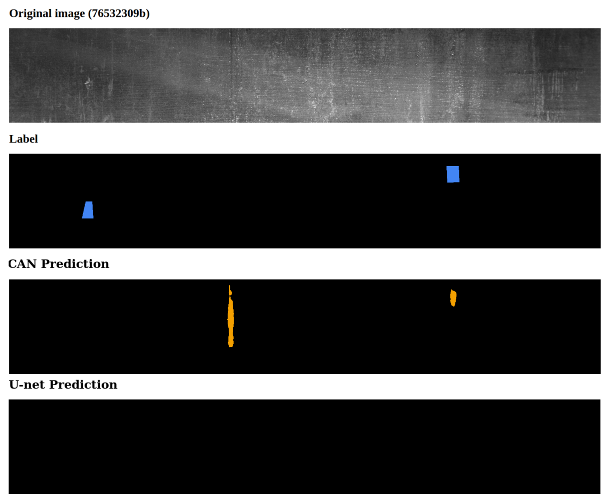
[13] G. P. F. S. H. A. Liang-Chieh Chen, Yukun Zhu, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv preprint arXiv:1802.02611v3*, 2018.



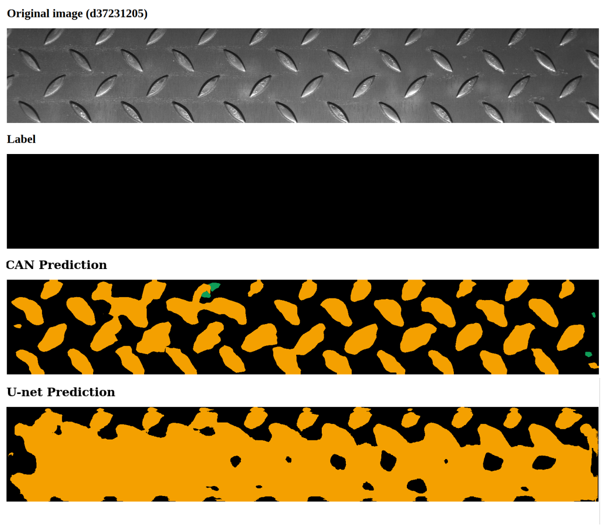
(a) An example of the prediction of CAN and U-net. Yellow: defect type 3, Green: defect type 4.



(b) Another example of the prediction of CAN and U-net. Red: defect type 2.



(c) A fail case due to class imbalance. Blue: defect type 1



(d) A fail case of falsely recognizing patterns as defects.

Figure 4: Qualitative results of CAN. For each figure, up: original, middle: ground truth, bottom: prediction of model.