Automated Detection of Steel Defects via Deep learning

Neeraj Katiyar, Mohammad Farzanullah, Arish Yaseen, Farhan Bishe

Electrical and Computer Engineering Department, McGill University.

Abstract

Motivation: ***In the steel industry, there has been emphasis on the use of deep learning techniques to detect defects. This would increase the efficiency, improve automation, and will lead to high quality production of steel. In the traditional production process of steel materials, localizing and classifying surface defects manually on a steel sheet is inefficient and error prone. Therefore, it's a key challenge to achieve automated detection of steel surface defects in image pixel level, leaving an urgent and critical issue to be addressed. In this paper, we aim to increase the efficiency and enhancing the automation in detecting steel defects while maintaining the high quality in steel production by applying a series of deep learning algorithms of real-time semantic segmentation, utilizing neural networks with encoder-decoder architectures based on U-Net and feature pyramid network (FPN). The image dataset of steel defects is provided by Severstal, the largest steel company in Russia, through a featured code competition in the Kaggle community.***

Results: ***The results show that the ensemble algorithm of several neural networks with encoder decoder architectures has a decent performance regarding segmentation accuracy. Our machine learning algorithms achieve dice coefficients of 0.89901 and 0.90350 on the private test set and public test set on the Kaggle platform, respectively, which locates at the top 10% among all teams in the competition.***

Keywords—Deep Learning, U-NET, FPN, Steel Defect Detection, Semantic Segmentation.

1 Introduction

Steel is considered a core material and plays a vital role in modern day architectural structures by providing high strength and resistance to natural and man-made wear. The process of manufacturing steel sheets is complex as well as very delicate, as it involves multiple processes such as heating, rolling, drying, and cutting. During the whole process many heavy industrial machines encounter the thin steel sheet, making it prone to many kinds of defect, before it can be shipped. It is necessary to localize and classify these defects in time to prevent any catastrophic events later, and doing so manually on the fly is inefficient, erroneous and highly dangerous to the worker (Amin and Akhter, 2020). The steel manufacturing process can be made much more efficient with help of machine learning and image-based classification techniques, such as image segmentation to identify, localize and classify different kinds of defects in steel sheets with high accuracy. Therefore, achieving better efficiency, more automation and consistent high quality in the manufacturing process. To help make production of steel more efficient, we propose a machine learning based semantic segmentation algorithm instead of manual detection method. This will help localize and identify surface defects on a steel sheet, which can also improve automation, increase efficiency, and maintain high quality in manufacturing process.

The scope of defect detection in steels was very limited for researchers due to the lack of large-scale and high-quality image data of steel defects (Qian, 2019). Some of the existing well-known datasets such as NEU, UCI (University of California Irvine) steel plate fault datasets contain a very small quantity of training samples which cannot provide much useful information. Considering the limitations, the researchers can only explore the traditional machine learning approaches such as random forest, decision trees, and support vector machines etc. instead of potential deep learning techniques.

Severstal, a Russia based steel company, has recently generated, and published a high-quality, industry level steel image dataset, which contains pixel-wise annotation mask labels. Severstal has hosted a featured deep learning competition on the Kaggle platform facilitating this large-scale and high-quality dataset. The amin of the competition is to power the largest community of machine learning practitioners and data scientists to innovate novel solutions for industry-applications. In this study, we have explored the application of modern deep learning techniques based on deep neural network with encoder-decoder architecture leveraging transfer learning concepts. With the combination of Unet and feature pyramid network (FPN) with advanced pretrained classification network we achieved the automation detection of defects using real-time image semantic segmentation. Specifically, our deep learning algorithms has obtained coefficients over 0.915 and 0.905 at a speed of 1.5 images per second on the public test set and private test set on the Kaggle platform respectively, which locate us among the top 10% of all teams in the competition.

1.1 Data analysis

The dataset was obtained from a Kaggle competition 'Severstal: Steel Defect Detection' (Severstal, 2019). We have been provided with 12568 training images and 5506 test images. Each image is a grayscale image of dimensions 1600x256. On performing data analysis, we realized that there are 6666 images with defects and 5902 has at least one defect. Figure 1 shows the number of images with each of the defect and Figure 2 shows the number of defects in each image. There are a maximum of 2 defects in an image. Some of the sample images are displayed in Figure 3.

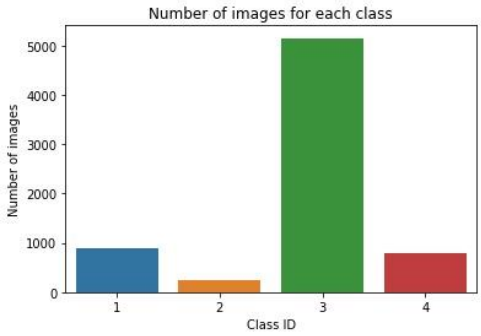


Figure 1. Distribution of images in four defects

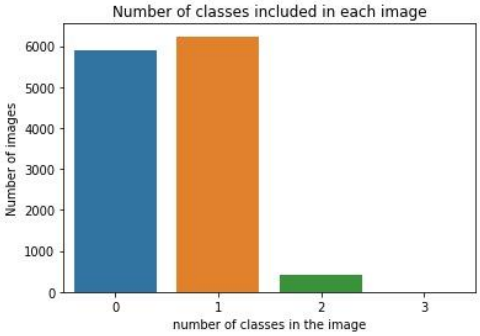


Figure 2. Distribution of defects in each image

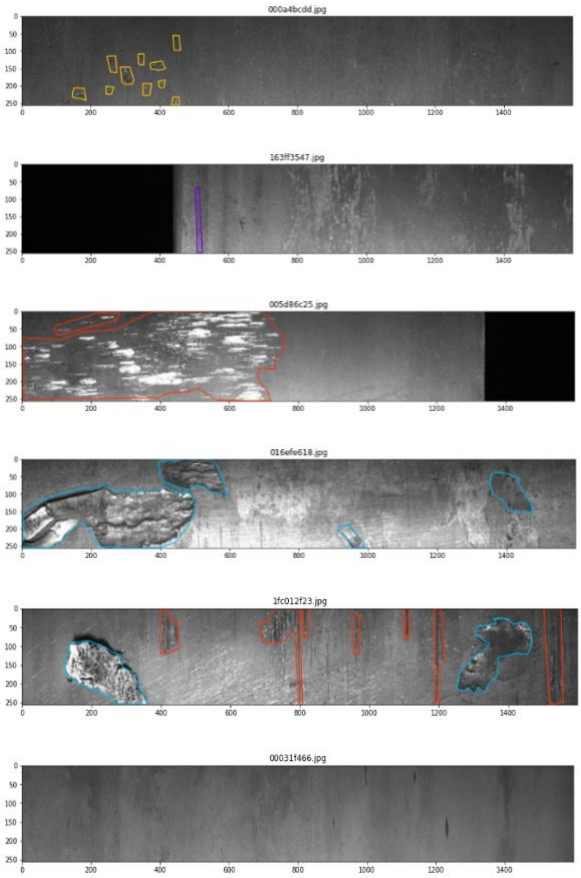


Figure 3. Sample original images

2 Methods

2.1 Semantic Segmentation Models:

2.1.1 U-Net

U-Net architecture has been proposed by (Ronneberger *et al.*, 2015) to do the image semantic segmentation task in a more efficient way. The conventional deep neural networks used for doing image classification are not able to localize each image's segment, while in some important applications (e.g., biomedical image processing, steel defect detection etc.) it is essential to know where each segment is precisely located. As a result, instead of assigning the class label to the whole image we need to assign it to each image pixel. Moreover, this structure addresses the problem of small datasets by doing a strong data augmentation. The U-Net structure consists of two paths: 1) Contraction path 2) Expansion path. Through contraction path the context of each image is captured and after that accurate localization is performed via expanding path. On the contraction path, the input image is passed through two three by three convolution and after that two by two max pooling is done. This procedure is repeated for three times until we reach to the bottom of the architecture. On the expansion path the reverse procedure is done by two by two up-convolution. The architecture of U-Net is shown in Figure 4.

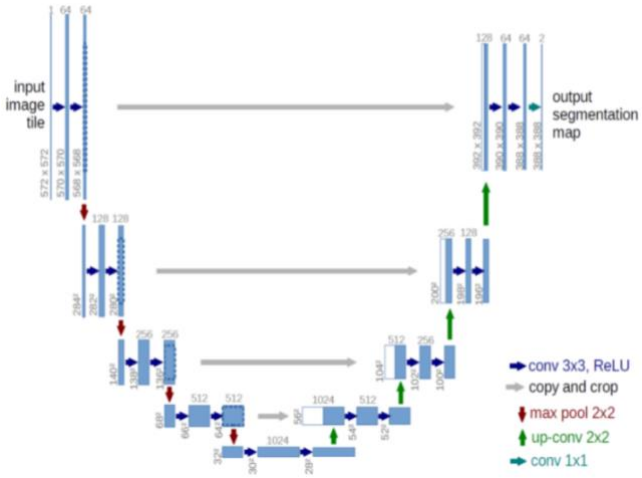


Figure 4. The schematic diagram of U-Net architecture (example for 32x32 pixels in the lowest resolution)

2.1.2 Feature Pyramid Network

Feature pyramid network (FPN) is an architecture being used for object recognition and semantic segmentation purposes at different scales. This structure has been proposed by Facebook AI Research (FAIR) (Lin *et al.*, 2017). This model performs both instance and semantic segmentation through a top-down structure with lateral connections. In this approach, the input is an image with an arbitrary size, and at each level of the FPN output is an image with the corresponding scale. The bottom-up pathway is constructed via typical feed forward convolutional neural network. The higher levels of this path correspond to the low-resolution but semantically strong features. In order to obtain high-resolution features from semantically strong feature levels, FPN utilizes the top-down path. The features

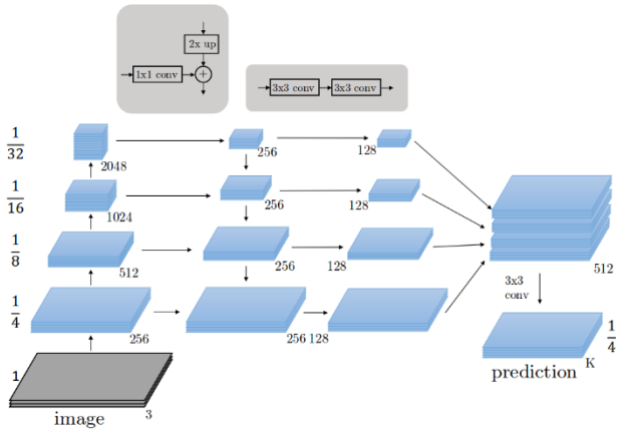


Figure 5. The schematic diagram of FPN architecture

on the top-down path can be enhanced by the corresponding features on the bottom-up path. Consequently, to perform a precise localization FPN exploits the lateral connection between the bottom-up path and the top-down path. The FPN structure is shown in Figure 5.

2.1.3 Deep Lab

Deeplab is another popular semantic segmentation architecture. Deeplab is a model designed and open sourced by Google (Chen *et al.*, 2014). It uses Atrous convolution in place of deconvolutional networks. Atrous convolution allows to enlarge the field of view of filters without increasing the number of parameters or the amount of computation. Multiple atrous convolutions are used in parallel to catch the contextual information at multiple scales. We used deeplab v3plus which is an upgrade to the previous versions and improves the results across object boundaries. The deeplab v3plus obtained an accuracy of 89% on PASCAL VOC 2012 test datasets (Chen *et al.*, 2018).

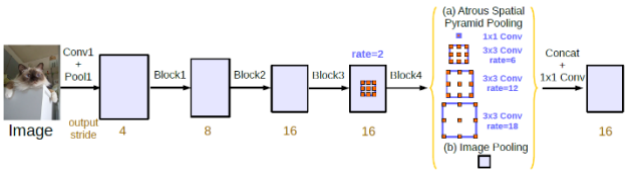


Figure 6. The schematic diagram of DeepLab architecture

2.2 Encoder Architecture

The discussed models in previous section utilize simple encoders which have not sufficient weight parameters resulting in ordinary performance. As a result, more efficient encoders can be designed through designing powerful convolutional neural networks to enhance the performance of the model. We have used the model encoders pretrained on the ImageNet dataset. Here, we give a brief explanation for each encoder that we have used for our models.

2.4.1 ResNet

The idea of "Skip Connections" was first introduced in ResNet by (He *et al.*, 2016). Implementing very deep neural network with hundreds of layers can lead to vanishing gradient issue which can stop the training procedure. By the emergence of the ResNet architecture, training extremely deep neural networks without encountering vanishing gradient issue became possible.

2.4.2 SE-ResNet

SE-ResNet encoder is the combination of Squeeze-and-Excitation network (SENet) and ResNet architecture introduced by (Hu *et al.*, 2018). In comparison with ResNet, instead of skipping only one path, ResNet can skip a group of paths via a skip connection. The number of paths that can be skipped by ResNet is defined as cardinality dimension. Considerable performance enhancement can be achieved by augmenting SE-Net to ResNet architecture.

2.4.3 EfficientNet

This architecture deals with properly scaling the convolutional neural networks. Each CNN can be scaled in three different ways: 1) Network depth 2) Network width 3) Resolution where depth corresponds to the number of layers, width corresponds to number of units per layer and the resolution corresponds to the image size. In order to optimize scaling the CNN in all three areas Google has proposed EfficientNet structure which can enhance the accuracy and efficiency of the CNNs (Tan and Le, 2019).

2.3 Data Processing

For the preprocessing step, we have normalized the images using the mean and standard deviation of the ImageNet images. Moreover, we have augmented the dataset by applying vertical and horizontal flip both with probability of 0.5, random brightness, random contrast, and random hue in range of [0.8,1.2] of the actual image. Besides, we have applied Gaussian noise with the probability of 0.5. We trained 21 different semantic segmentation models which are discussed in Section 3.2. While making predictions for these models, the defect masks containing less than 3500 pixels are ignored since they are mostly false positives. Also, the threshold of 0.5 is set for all models to see whether a pixel is defected or not.

2.4 Training methodology

We have used transfer learning procedure to train our models which resulted in much better performance in comparison with the models trained without transfer learning. In this work we have trained different models. The information regarding each model and the corresponding results are provided under section 3.2.

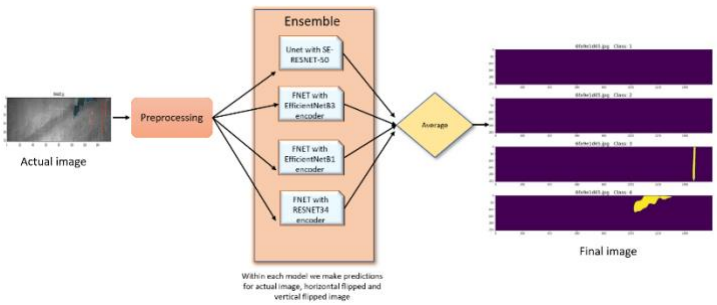


Figure 7. High level workflow of end-to-end learning and detection process

2.5 Final Model

Our final model is an ensemble model for segmentation tasks, which is composed of U-Net with an encoder of SE-Resnet-50 and PFN with encoders EfficientNet-B1, EfficientNet-B3 and ResNet-34. We use soft voting method to ensemble these models and then use a threshold postprocess module to filter tiny masks (which are mainly false positive predictions). For each test image, for each model, the actual image, horizontally flipped image and vertically flipped image is predicted. Consequently, as we make use of four models in our final model each image is predicted 12 times. Afterward, the sigmoid values of these 12 outputs are averaged. Each pixel is classified as defect if its value is greater than the threshold value. Threshold values obtained after extensive trial and error are $0.5, 0.6, 0.45, 0.4$ for the four defect classes. Moreover, if the defect's mask size is less than minimum size (1400,1800,1200,1300) then the mask is dropped. Final prediction is generated by such ensemble model in the format of run-length encodings (RLE) since we need to submit the RLE for each image and class combination to Kaggle.

3 Results

3.1 Evaluation metric

The Kaggle competition uses the Dice Coefficient as the evaluation metric. This metric can provide the pixel-wise comparison between the predicted segmentation and the true values. The formula is given as follows:

$$Dice = 2 \times \frac{|X \cap Y|}{|X| + |Y|}$$

where X and Y are predicted pixels and true pixels, respectively. $|X|$ represents the total number of pixels in $|X|$, $|Y|$ represents the total number of pixels in Y, and $|X \cap Y|$ represents the total number of pixels which are common in X and Y. If X and Y are empty the Dice coefficient will be equal to 1. Kaggle calculates the score as the mean of the dice coefficients of all the pairs of image and defect classes.

3.2 Comparison of semantic segmentation of models

We trained 21 different semantic segmentation models and encoder combinations to compare and find the best models. For the comparison, all models were trained using Binary Cross Entropy Loss with Adam Optimizer with a learning rate of 0.0005. The data was divided into training and validation with a ratio of 80-20 keeping the seed fixed. The learning rate was scheduled such that it decreased by a factor of 0.1 if the validation loss did not decrease for 3 consecutive epochs. The number of epochs were set as 20. All the models used the same post processing to make predictions on Kaggle. The post processing included to remove any mask from the predicted mask that is less than 3500 pixels. Moreover, the threshold for defect was set at 0.5. The results are indicated in Table 1 and compares the score on Kaggle's private and public leaderboard.

The result indicates that the model that gave the highest score on private dataset was UNet with an encoder backbone of se-resnet-50. This was followed by FPN with EfficientNet-b3 and EfficientNet-b1 as encoders. Other semantic segmentation models such as Multi-Attention-Network (MANet), Pyramid Attention Network (PAN) and LinkNet were unable to perform well. The score of 0.89103 would correspond to 445th place at Kaggle. For further improvement we considered hyperparameter optimization and ensemble techniques.

Architecture	Encoder	Public	Private
U-Net	se_resnet_50	0.88918	0.89103
FPN	EfficientNet-b3	0.88777	0.8891
FPN	EfficientNet-b1	0.89629	0.88865
DeepLabV3Plus	ResNet34	0.88629	0.88851
FPN	ResNet34	0.8877	0.88825
U-Net	se_resnext_32	0.88695	0.88711
U-Net	ResNet34	0.88312	0.88701
U-Net	EfficientNet-b2	0.88868	0.88663
UNetPlusPlus	ResNet34	0.88386	0.8854
U-Net	DenseNet121	0.88494	0.88529
FPN	ResNet18	0.88508	0.88393
U-Net	ResNet18	0.88689	0.8836
U-Net	VGG11	0.88434	0.88259
U-Net	EfficientNet-b1	0.87904	0.88211
U-Net	EfficientNet-b3	0.88274	0.88168
FPN	se_resnet_50	0.87911	0.88159
PAN	ResNet34	0.88066	0.8801
LinkNet	ResNet34	0.88416	0.87972
FPN	EfficientNet-b2	0.88502	0.87894
MANet	ResNet34	0.87473	0.87764
U-Net	Inceptionv4	0.88234	0.87509

Table 1. Results from various semantic segmentation models

3.3 Hyperparameter Optimization

For hyperparameter optimization, we first tried to use other loss functions on our models. We trained using the BCE-Dice loss on U-Net with Resnet34 encoder and obtained the private score of 0.88605 which was lower than the score using BCE loss of 0.88701. Moreover, we trained the U-Net with Resnet18 encoder using Focal Loss and obtained private score of 0.85636 which was lower than the score using BCE loss of 0.8836. It was concluded that the most suitable loss for our problem is BCE loss. Furthermore, for our best model of U-Net with se-resnet-50 encoder, we tried different combination of optimizers, learning rate and weight decay which are summarized in the table below:

Optimizer	Learning Rate	Weight Decay	Public	Private
Adam	5.00E-04	0.00E+00	0.88918	0.89103
RMSProp	5.00E-04	0.00E+00	0.88493	0.88797
SGD	5.00E-04	0.00E+00	0.85674	0.8556
Adam	1.00E-03	0.00E+00	0.88741	0.88873
Adam	5.00E-04	1.00E-04	0.87667	0.87462

Table 2. Hyperparameter optimization performed on U-Net with SE-ResNer-34 encoder

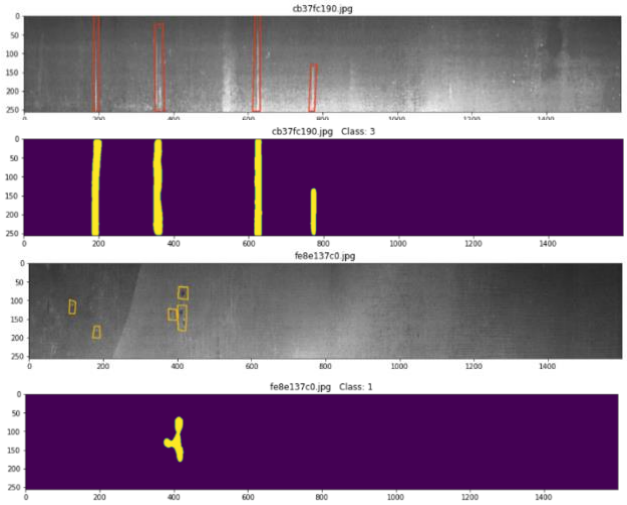
3.4 Ensemble results

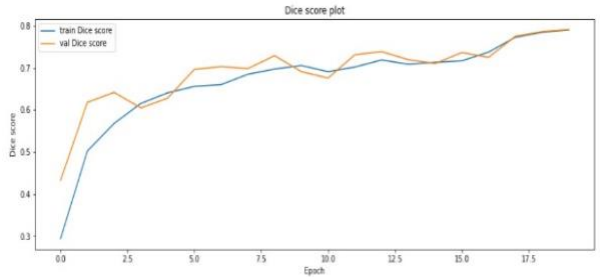
The details for the ensemble model are shown in section 2.5. The ensemble technique improved our score to 0.89901 on the private leaderboard which corresponds to 198th position (8.2%). The comparison of our technique with the winner of the competition and a baseline are presented below.

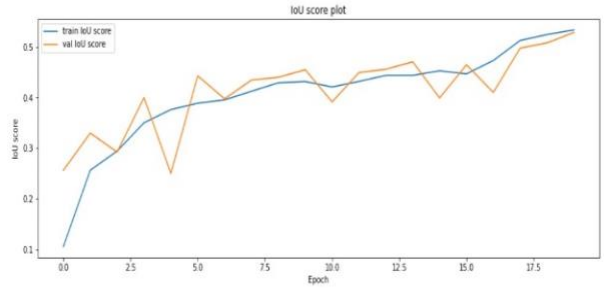
Model	Private
Our ensemble model	0.89901
Kaggle's winner model	0.90883
No defects prediction	0.8556

Table 3. Result comparison of winner's model and our best model

The figure compares the actual defects with the defects predicted by our model. It can be seen that our model performs reasonably well on segmenting the defects from the steel image.







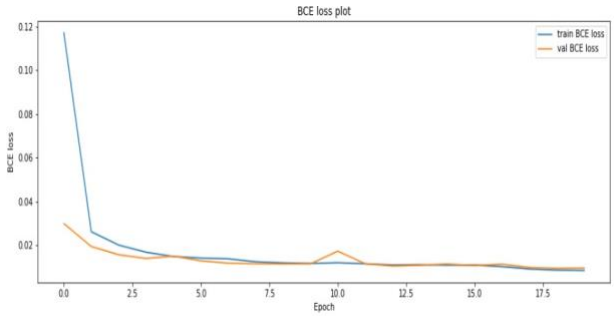


Figure 8. Actual defects vs detected defects

3.4.2 Performance plots

Following plots shows the Dice score, IoU score and BCE loss over epoch.
Figure 8 (a) Dice score of the best model (b) IoU score of the best model (c) BCE loss of the best model

4 Discussion and Conclusions

We were able to develop an efficient model for semantic segmentation by ensemble of 4 models consisting of U-Net and FPN architectures. This model gave an accuracy of 0.89901 and 0.90350 on private and public leaderboard on Kaggle, respectively. This was in the top 8.2% of the competitors. Our model can be a useful tool for detecting steel defects in industry to identify and localize the defects swiftly and accurately. Within the individual models, we observed that U-Net and FPN performed better than other semantic segmentation models. Moreover, better results were generally obtained using ResNet and EfficientNet encoders.

In the future, we can further improve the score by using pseudo-labeling technique. We will use the images of the test set that were predicted with high confidence as part of our training data and fine tune our models. Furthermore, more efficient models can be designed that take less time for training and inference.

References

Chen,L.-C. *et al.* (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In, *Proceedings of the European conference on computer vision (ECCV)*., pp. 801–818.

Chen,L.-C. *et al.* (2014) Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv Prepr. arXiv1412.7062*.

He,K. *et al.* (2016) Deep residual learning for image recognition. In, *Proceedings of the IEEE conference on computer vision and pattern recognition*., pp. 770–778.

Hu,J. *et al.* (2018) Squeeze-and-excitation networks. In, *Proceedings of the IEEE conference on computer vision and pattern recognition*., pp. 7132–7141.

Lin,T.-Y. *et al.* (2017) Feature pyramid networks for object detection. In, *Proceedings of the IEEE conference on computer vision and pattern recognition*., pp. 2117–2125.

Ronneberger,O. *et al.* (2015) U-net: Convolutional networks for biomedical image segmentation. In, *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Severstal (2019) Severstal: Steel Defect Detection. *Kaggle*.

Tan,M. and Le,Q. (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In, *International Conference on Machine Learning*. PMLR, pp. 6105–6114.