

Surface Defect Detection Using YOLO Network

Abstract. Detecting defects on surfaces such as steel, can be a challenging task because defects have complex and unique features. These defects happen in many production lines and differ between each one of these production lines. In order to detect these defects, the You Only Look Once (YOLO) detector which uses a Convolutional Neural Network (CNN), is used and received only minor modifications. YOLO is trained and tested on a dataset containing six kinds of defects to achieve accurate detection and classification. The network can also obtain the coordinates of the detected bounding boxes, giving the size and location of the detected defects. Since manual defect detection is expensive, labor-intensive and inefficient, this paper contributes to the sophistication and improvement of manufacturing processes. This system can be installed on chipsets and deployed to a factory line to greatly improve quality control and be part of smart internet of things (IoT) based factories in the future. YOLO achieves a respectable 70.66% mean average precision (mAP) despite the small dataset and minor modifications to the network.

Keywords: YOLO, Defect Detection, CNN, Computer Vision, Transfer Learning.

1 Introduction

In every factory, defects can occur on products rolling out at the end of the conveyor line. This is due to many factors such as contamination, human error, machinery malfunctions and more. These defects include scratches and patches and not only is the defect purely cosmetic, in some cases it is structural and can cause damage to the steel surface such as corrosion, low wear resistance and short fatigue life which can lead to disastrous results where the products are meant to be used [1]. In order to show the importance of catching steel surface defects, tests are conducted on structural steel with and without defects and the results showed that metal surfaces with defects have 40% less strength with much faster strength degradation [2]. Safety is another very important factor to consider since metal surfaces are used in all kinds of applications ranging from automotive applications all the way to construction.

To keep up with the production lines requirements, the designed defect detector must be accurate and fast. Factories these days have come a long way and work at a very high pace rolling hundreds of products out every hour. The detector must also be

able to distinguish between defects and non-defective interference such as dust. Inspection and quality assessment used to be done manually by humans who are prone to suffer from exhaustion and can be slower than machines. Moreover, training operators requires time and money and finding people fit for the job is not easy in the first place and as mentioned earlier the usage of steel plates ranges over a large number of applications with some being critical and dangerous in the case of defects not being caught. Computer vision is helping in visual inspection and replacing manual labor in many industries [3].

CNNs are one of the best options for computer vision tasks. CNNs have allowed many advances in applications like image segmentation [4], [5] and the classification of objects [6], [7]. CNNs have also been used in industrial applications [8], [9], [10]. Moreover, CNNs have convolution layers that take care of feature extraction, they are rugged when it comes to shifts and distortions in the image, they require less memory and the training is easier and they are better and faster due to the reduced number of parameters.

In this paper, the YOLO network is used for the detection and classification of various defects in steel surfaces. The network is also able to extract the coordinates of the defects which in return gives the location and size of each detected defect.

This paper is structured as follows: In section 2, the background is presented, in section 3, the methodology is explained including the training and testing process, section 4 contains a discussion and analysis of the results and finally, section 5 concludes this paper and mentions future work.

2 Background

2.1 YOLOV3 and Darknet-53

YOLO [11] is a one-shot object detection algorithm and it is one of the fastest algorithms that exist today. It is mostly used in areas where speed is a crucial element without the loss of too much accuracy. It uses a convolutional neural network which is crucial when it comes to feature extraction. The way YOLO works is it divides an image into an $S \times S$ grid of cells where each cell is then responsible for predicting if there is an object in it, $P(\text{object})$, as well as producing a number of bounding boxes which are likely to encompass objects. Each of the predicted bounding boxes has a confidence score where confidence is:

$$P(\text{Object}) \times \text{IOU}(\text{pred}, \text{truth})$$

Intersection Over Union (IOU) is used to measure the difference between the ground truth bounding box and the predicted bounding box (Fig 1). The predicted bounding boxes that are closest to the ground truth are kept and their confidence scores are increased whereas the boxes that have a low IOU intersection with the ground truth are given a low confidence score. Five values are predicted by the network for each bounding box. (x,y) are the center of the bounding box and (w,h) are the width and height.

The next prediction is the conditional probability, $P(\text{Class}|\text{Object})$, where the probability of a certain class being in one of the bounding boxes is calculated. The final predictions are many bounding boxes scattered all around the image. YOLO thresholds the detections using non-maximum suppression (NMS) to remove unwanted and duplicate bounding boxes. The network then ends up with only the necessary predictions shown on the image (Fig 2).

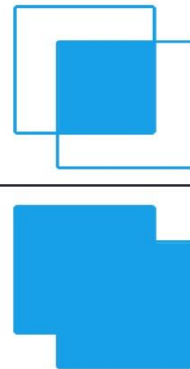
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Fig. 1. Intersection Over Union [12]

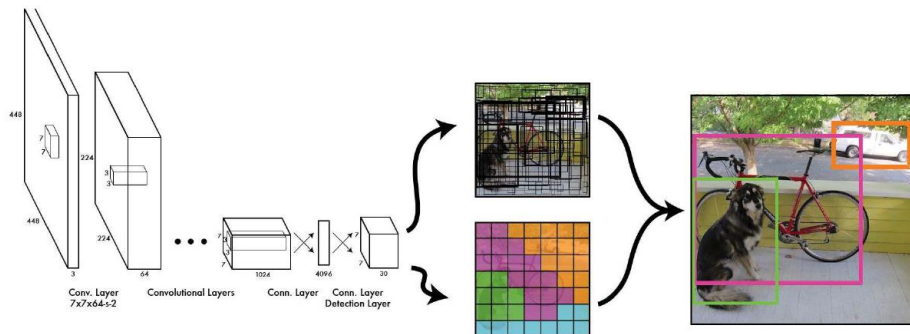


Fig. 2. YOLO Network Pipeline [13]

The backbone of YOLO is called DARKNET [14] created by Joseph Redmon, which is a neural network framework written in NVIDIA's Compute Unified Device Architecture (CUDA) and C. Its advantages are that it is quick, slim and easy to work with. Unlike its predecessor, YOLOv3 uses DARKNET-53 instead of DARKNET-19. DARKNET-53 has 53 convolutional layers trained on ImageNet, is much deeper than the previous versions. It composes mainly of 3×3 and 1×1 filters with shortcut connections. It is also faster due to better utilization of the GPU. Darknet-53 is also proven to have better performance than ResNet-101 and it is 1.5 times faster and compared to ResNet152 it has similar performance but is 2 times faster [11]. DARKNET has its own commands and parameters which are used to train, test, calculate and perform many other operations on the model being worked on. This paper uses a slightly modified DARKNET53 by AlexeyAB [15] to allow for training on custom datasets.

2.2 Related Work

There are many methods for surface defect detection. In a paper [16], a simple CNN model is presented to detect defects in metal steel surfaces where the model achieved moderate results. However, with changes in the number of batches, as well as some data augmentation, 99% accuracy is achieved in training and testing. In [17], a two-layer convolutional network is proposed to detect surface defects where the loss function is calculated using categorical cross-entropy. After testing the system on testing images, the system is found to be 64.7% accurate which is acceptable given the small dataset. The disadvantage here is that there are only two convolutional layers which is not enough to extract features from a very small dataset. This issue is tackled in another paper [18] where it is decided to modify the YOLO detector to be fully convolutional where the network has 25 convolutional layers for feature extraction and 2 convolutional layers to predict the defect class and bounding box. With this architecture, the YOLO network is able to learn its own spatial downsampling instead of deterministic spatial downsampling. In this case, YOLO achieved a mAP of 97.55% and a recall rate of 95.86%. Another paper [19] which explored an approach for surface defect detection using deep learning used a two-stage method which comprised of a segmentation network and decision network. The model worked fine and better than other approaches when experimenting on the Kolektor Surface-Defect Dataset (KolektorSDD) however it still experienced 5 misclassifications and suffered a bit when it came to images with lower resolution. It did, however, achieve an accuracy of 99%. Table 1 shows a comparison of different approaches used to achieve the same goal as this paper with some of them using similar images and different performance metrics.

Table 1. Performance comparison of different methods

MODEL	PERFORMANCE MEASURE
CNN, Gathered dataset [16]	Acc: 99% mAP: N/A
2 Layer CNN, NEU surface defect database [17]	Acc: 64.7% mAP: N/A
Fully Convolutional YOLO, Gathered dataset [18]	Acc: N/A mAP: 97.55%
Segmentation + Decision Network, KolektorSDD [19]	Acc: 99% mAP: N/A

3 Methodology

Originally, YOLO is a pretrained object detector, trained to detect everyday objects such as tables, chairs, cars, phones and others. A modified version of YOLOV3 is used in this paper. Changes to the hyperparameters are made to be able to train and test using the custom dataset provided. The original dataset labels needed some modifications since YOLO only accepts a specific format and 5 specific parameters to associate the labels to the images and train properly.

3.1 Dataset

The images are obtained from the Northeastern University (NEU) surface database [20], [21], [22] which contains six types of defects (rolled-in scale (Rs), patches (Pa), crazing (Cr), pitted surface (Ps), inclusion (In) and scratches (Sc)) with 300 images for each defect (1800 total). Image size is 200×200 pixels with a .bmp format and the images are in grey-scale. The defects in the images vary and are provided in many shapes, sizes, illumination and orientation. The images are already labelled, and the labels contained information such as the location and size of the bounding box in an XML format. For this paper, the images are resized to 608×608 pixels using an online resizing tool [23] since the original size is too small. YOLO automatically resizes the input images to smaller dimensions when training, so it is crucial to start off with a somewhat large image so that the defects to be detected in the images are not too small, but closer in size to defects in images and videos provided by cameras in factories [24]. After many trials, it is found that 608×608 pixels is the best size and gave the best results. The labels are modified as well to fit the YOLO format since YOLO takes five values to produce the bounding boxes. Therefore, the results are 1800 text files each containing the five values in the following format: “(object-id) (x-centre) (y-centre) (width) (height)”. The images are split into 10% for testing and 90% for training. It is important to note that data augmentation is not used in this paper on purpose in order to show that YOLO achieved good results with limited data.

3.2 Google Colab

All training and testing tasks are performed using a 12GB NVIDIA Tesla K80 GPU provided by Google Colab which is compatible with DARKNET since, as mentioned earlier, DARKNET is written in C and CUDA. The NVIDIA CUDA deep neural network library (cuDNN) is used to make it all work.

3.3 Training and Testing

Since DARKNET-53 is pretrained, transfer learning is used to train YOLO on the NEU dataset. When training an object detector, it is always good to start from an existing model trained on very large datasets and then use the weights of this model to train. This is fine even if the trained weights do not contain the objects required in this experiment. This process is called transfer learning. A pretrained model that contains

weights trained on ImageNet is used as starting weights so that the network can learn quicker. This is also beneficial since fewer data will be required [25] which is convenient since the NEU dataset only has 300 images per class before train/test split.

Several parameters are changed in order to train and test YOLOV3 using a custom dataset. However, one of the goals of this paper is to achieve this with very minor modifications to the network. Which is why most of the parameters are left the same way they came with YOLOV3. Some of the unchanged parameters include the loss function where YOLOV3 uses the sum-squared error in the loss function and to maximize the efficiency of this function, the network increases the confidence score as much as possible for it to be equal to the IOU between the ground truth and the predicted bounding box and decreases the confidence score when there are no objects in the bounding box.

Another intact parameter is the activation function. YOLOV3 uses leaky activation function for each convolutional layer except the last one before each YOLO layer where a linear function is used. The linear activation function is also used in the shortcut connections.

At first, YOLO training reached an average loss of 0.11 which is supposed to be good, however the network did not converge, the mAP was very low, no detections were made even on training images, true positive and false positive values were almost null and finally, the accuracy for each class was mostly 0.00% which meant more research and changes had to be made in order to get better results (Figure 3).

```
calculation mAP (mean average precision)...
164
detections_count = 55, unique_truth_count = 391
class_id = 0, name = Crazying, ap = 0.00%      (TP = 0, FP = 0)
class_id = 1, name = Inclusion, ap = 0.00%      (TP = 0, FP = 0)
class_id = 2, name = Patches, ap = 0.00%      (TP = 0, FP = 4)
class_id = 3, name = PittedSurface, ap = 0.00% (TP = 0, FP = 2)
class_id = 4, name = RolledInScale, ap = 0.17% (TP = 0, FP = 0)
class_id = 5, name = Scratches, ap = 0.00%    (TP = 0, FP = 1)

for thresh = 0.25, precision = 0.00, recall = 0.00, F1-score = -nan
for thresh = 0.25, TP = 0, FP = 7, FN = 391, average IoU = 0.00 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.000291, or 0.03 %
Total Detection Time: 11.000000 Seconds
```

Fig. 3. Results with 0.03% mAP

The next attempt was to troubleshoot the problem so 4 classes were removed, and then YOLO was left to train for only two classes which had defects easy to detect. After training, the results obtained were about the same, however, YOLO was able to detect defects in some images with very low confidence. But obviously, the results were not good considering only two classes were used.

Another trial was attempted where YOLO was left to train for more iterations and the results barely improved. This meant that the number of iterations was not the cause of bad results.

The dataset images were 200x200 in size however the network size was 416x416. YOLO has a built-in feature which allows it to resize images on its own in order to get the best out of the training however it was later discovered that this was not working properly since the network size was bigger than the images. After experimenting with resizing the images and resizing the network it was concluded that with a network size of 416x416 and image size of 608x608 the network achieved the best results so far (figure 4).

```

calculation mAP (mean average precision)...
164
detections_count = 922, unique_truth_count = 391
class_id = 0, name = Crazying, ap = 0.24% (TP = 0, FP = 0)
class_id = 1, name = Inclusion, ap = 25.88% (TP = 14, FP = 5)
class_id = 2, name = Patches, ap = 36.71% (TP = 29, FP = 21)
class_id = 3, name = PittedSurface, ap = 1.62% (TP = 0, FP = 1)
class_id = 4, name = RolledInScale, ap = 9.11% (TP = 4, FP = 2)
class_id = 5, name = Scratches, ap = 47.71% (TP = 12, FP = 4)

for thresh = 0.25, precision = 0.64, recall = 0.15, F1-score = 0.24
for thresh = 0.25, TP = 59, FP = 33, FN = 332, average IoU = 46.04 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.202104, or 20.21 %
Total Detection Time: 11.000000 Seconds

```

Fig. 4. Results with 20.21% mAP

A Mini-batch gradient descent is used where a certain number of batches is taken during training. A Mini-batch gradient descent finds a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. Smaller batch sizes are noisy, offering a regularizing effect and lower generalization error and makes it easier to fit one batch worth of data in memory. The number of batches is lowered from 64 to 24 and the subdivisions from 64 to 8. The use of small batches as opposed to the typical use of large mini-batches, is proved to get better generalization and allows for a smaller memory footprint [26]. This is by far the most affecting factor in this experiment. The results improved significantly, and the network can converge and detect defects in all images including test images with high confidence.

4 Results and Analysis

In the final attempt, YOLO is trained for approximately 25000 iterations using the six types of defect images. As mentioned earlier, the batch number is lowered from 64 to 24 which helped raise the mAP. The learning rate is expected to start off high and then drop as the network learns and has more information and therefore requires less aggressive learning. This is exactly what happened, however at the beginning, the learning rate increased before reaching the point where it should decrease. This is called the burn-in period or the warmup period. Training took about 55 hours on the single 12GB NVIDIA Tesla GPU.

YOLO successfully made accurate detections and classifications on the test images provided with each detection taking up to an average of 85 milliseconds. The network achieved a mAP of 70.66%, 79% precision and 68% recall. The results can be seen in Fig 5. Some sample detections are shown in Fig 7 where it can be seen how YOLO detects, localizes and classifies each of the six defects by drawing a bounding box around the defect and displaying the percentage of confidence as well as the time it took for detection.

```

calculation mAP (mean average precision)...
164
detections_count = 848, unique_truth_count = 391
class_id = 0, name = Crazing, ap = 24.82%      (TP = 18, FP = 24)
class_id = 1, name = Inclusion, ap = 72.05%     (TP = 66, FP = 12)
class_id = 2, name = Patches, ap = 84.89%     (TP = 65, FP = 6)
class_id = 3, name = PittedSurface, ap = 87.79% (TP = 32, FP = 3)
class_id = 4, name = RolledInScale, ap = 62.31% (TP = 34, FP = 19)
class_id = 5, name = Scratches, ap = 92.09%   (TP = 50, FP = 5)

for thresh = 0.25, precision = 0.79, recall = 0.68, F1-score = 0.73
for thresh = 0.25, TP = 265, FP = 69, FN = 126, average IoU = 60.42 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.706573, or 70.66 %

```

Fig. 5. Final detection results

As an example, in the image containing a pitted surface defect, YOLO drew a bounding box around what it thinks is a pitted surface defect. It is 99% confident that the defect is correctly classified, and it took only 81.86 milliseconds for the whole process to be done.

The proposed model in this paper can also extract the coordinates of the resulting bounding boxes which in return allows obtaining the position of the defects as well as their sizes. The network outputs the coordinates to a text file, along with the name and accuracy for each of the defects detected. Fig 6 shows a detection of a metal sheet suffering from many scratches with the network giving the accuracy for each scratch as well as the center coordinates, height and width of the bounding box enclosing the scratches. The model can also make predictions in a matter of milliseconds and can be deployed on mobile devices such as cameras to be used in production lines since it is considered lightweight and can perform fast detections on just about any regular laptop. It can smoothly track and detect defects and it is robust enough when it comes to changes in size and orientation.

```

seen 64
Enter Image Path: /content/darknet/img/scratches_30.jpg; Predicted in 87.295000 milli-seconds.
Scratches: 83% (left_x: 37 top_y: 10 width: 87 height: 205)
Scratches: 55% (left_x: 184 top_y: -3 width: 73 height: 84)
Scratches: 75% (left_x: 258 top_y: 259 width: 82 height: 339)
Scratches: 53% (left_x: 385 top_y: 34 width: 59 height: 273)
Scratches: 71% (left_x: 391 top_y: 453 width: 46 height: 160)
Enter Image Path:

```

Fig. 6. Coordinate Extraction

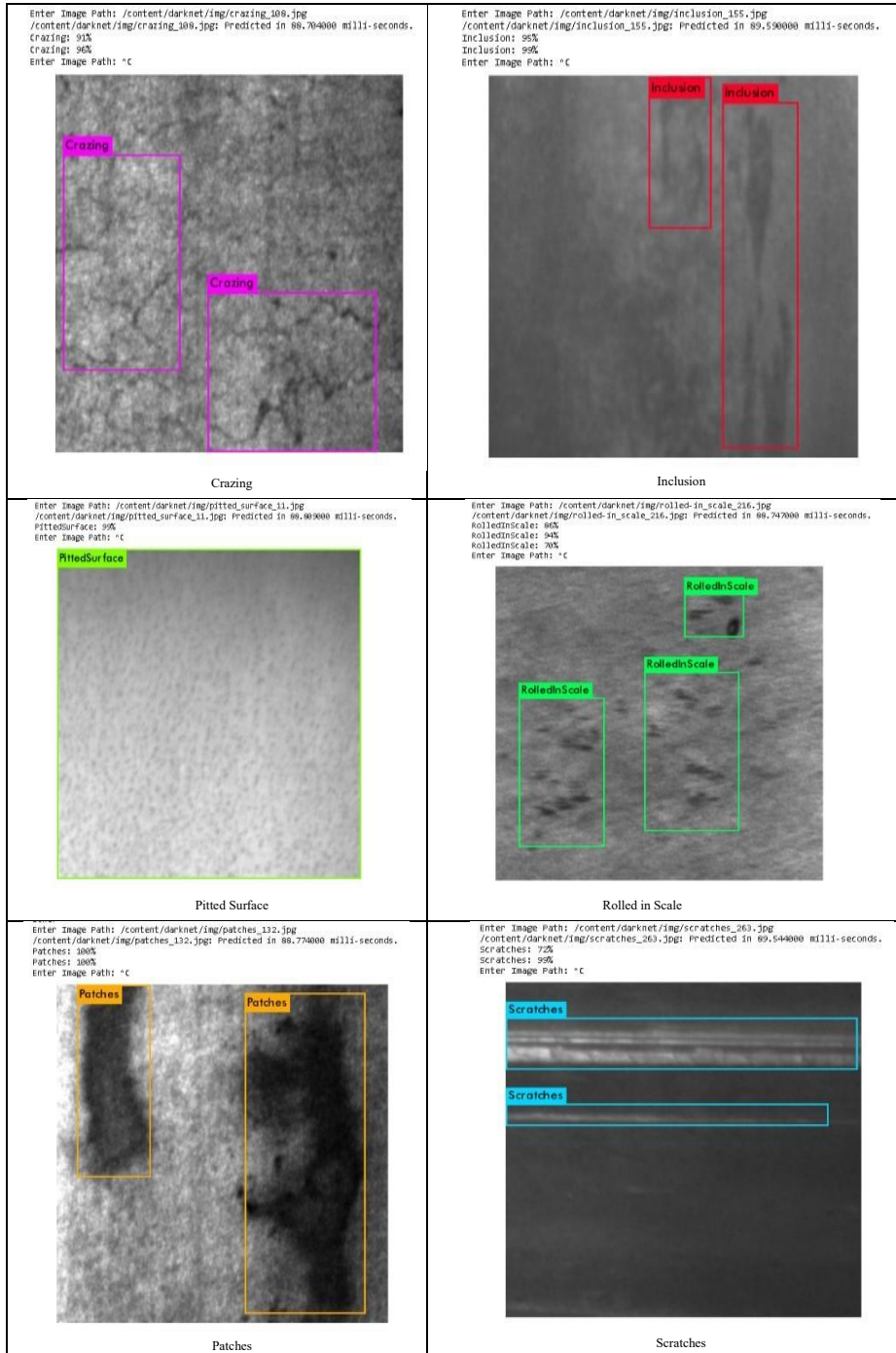


Fig.7. Sample Detections

Although YOLO can make detections and classifications correctly, one of the defects, crazing, has an average precision (AP) of 24% as opposed to the high AP that the other defects have. This is return caused the mAP to drop. Even with a lower AP, the network is still able to detect and classify crazing defects and with high accuracy.

The other papers mentioned in the background section, used either the NEU dataset [17] or preferred to gather their own images and labels or use known datasets [16], [18], [19] which are very similar to the NEU dataset images. Another thing to note is that a direct comparison of this paper with the other methods is not possible since most methods used have their own metric for measuring performance depending on the model used. In YOLOs case, the mAP is used. When it comes to detectors such as YOLO it is much better to use the mAP metric instead. The mAP has many advantages over other metrics like avoiding the “accuracy paradox” which is the accuracy increases even though the model is not actually good. This usually happens when True Positive (TP) < (False Positive) FP.

5 Conclusion

YOLOV3 detector is modified and then trained on a dataset containing six types of defects on steel surfaces. The dataset is prepared, and the labels configured to fit the YOLO format. After many trials and changes to the hyperparameters such as batch size and network size, YOLO is able to achieve a mAP of 70.66% with 79% precision and 68% recall. Most of the defects have high average precision with the exception of one which received 24% which in return affected the mAP. Despite this, the network still achieves accurate detections and classifications taking up to an average of 85 milliseconds. It must be noted as well that the results in this paper are obtained using a relatively small dataset with no data augmentation which is usually not enough to train a neural network or achieve decent results. The network also obtains the coordinates of resulting bounding boxes in order to calculate the size and position of the defects. This is important for improving the manufacturing process and the quality of products rolled out of factories. It is important to note that even though YOLO trained on metal steel surfaces, it can be used and trained on other surfaces such as wood, glass and paper.

Further work includes heavier modifications to the source files and hyperparameters such as learning rate, anchors, loss function and even altering the layers of the network by changing the values of filters and maybe adding or removing certain layers. Accuracy may also be improved in the case of a bigger dataset, pre-processing of the data and data augmentation techniques.

References

1. Sun, Q., Cai, J., Sun, Z.: Detection of Surface Defects on Steel Strips Based on Singular Value Decomposition of Digital Image. *Mathematical Problems in Engineering*, 1-12 (2016).
2. Jiang, Q., Sun, C., Liu, X., Hong, Y.: Very-high-cycle fatigue behavior of a structural steel with and without induced surface defects. *International Journal of Fatigue* 93, 352-362 (2016).
3. Neethu, N.J., Anoop, B.: Role of Computer Vision in Automatic Inspection Systems. *International Journal of Computer Applications* 123(13), 28-31 (2015).
4. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431-3440 (2015).
5. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(6), 1137-1149 (2017).
6. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60(6), 84-90 (2017).
7. Kaiming, H., Xiangyu, Z., Shaoqing, R., Jian, S.: Deep Residual Learning for Image Recognition, 770-778 (2016).
8. Masci, J., Meier, U., Ciresan, D., Schmidhuber, J., Fricout, G.: Steel defect classification with Max-Pooling Convolutional Neural Networks. *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-6. (2012).
9. Soukup, D., Huber, R.: Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images. *ISVC*, (2014).
10. Weimer, D., Scholz-Reiter, B., Shpitalni, M.: Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals* 65(1), 417-420 (2016).
11. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement 180402767, (2018).
12. Pyimagesearch Intersection over Union (IoU) for object detection, <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, last accessed 2020/01/24.
13. Redmon, J.: You Only Look Once: Unified, Real-Time Object Detection. Las Vegas, NV, (2016).
14. Redmon, J.: Darknet: Open Source Neural Networks in C. Pjreddie.com, (2019).
15. GitHub AlexeyAB/darknet, <https://github.com/AlexeyAB/darknet> , last accessed 2020/01/24.
16. Zhou, S., Chen, Y., Zhang, D., Xie, J., Zhou, Y.: Classification of surface defects on steel sheet using convolutional neural networks. *Materiali in tehnologije* 51(1), 123-131 (2017).
17. Islam, F., Rahman, M.: Metal Surface Defect Inspection through Deep Neural Network. 2018.
18. Li, J., Su, Z., Geng, J., Yin, Y.: Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-PapersOnLine* 51(21), 76-81 (2018).
19. Tabernik, D., Šela, S., Skvarč, J., Skočaj, D.: Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, (2019).

20. Song, K., Yan, Y.: A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science* 285, 858-864 (2013).
21. He, Y., Song, K., Meng, Q., Yan, Y.: An End-to-end Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. *IEEE Transactions on Instrumentation and Measurement*, 1-1, (2019).
22. He, Y., Song, K., Dong, H., Yan, Y.: Semi-supervised defect classification of steel surface based on multi-training and generative adversarial network. *Optics and Lasers in Engineering* 122, 294-302, (2019).
23. Bulkresizephotos Bulkresizephotos.com, last accessed 2020/01/24.
24. WINTRISS INSPECTION SOLUTIONS surface inspection, <http://www.winspection.com/surface-inspection.php>, last accessed 2020/01/24.
25. Aytar, Y., Zisserman, A.: Tabula rasa: Model transfer for object category detection. 2011 International Conference on Computer Vision. pp 2252-2259 (2011).
26. Masters, D., Luschi, C.: Revisiting Small Batch Training for Deep Neural Networks. ArXiv 180407612, (2018).