Automated surface defect detection in metals: a comparative review of object detection and semantic segmentation using deep learning

Rubén Usamentiaga*, *Senior Member, IEEE*, Darío G. Lema*, Oscar D. Pedrayes*, Daniel F. Garcia*

*Department of Computer Science and Engineering, University of Oviedo
33204 Gijón, Asturias, Spain Tel.: +34-985-182626, Email: rusamentiaga@uniovi.es

***Abstract*—Automated surface defect detection is a challenging problem that has attracted major attention for decades. Traditional methods were designed using a pipeline of carefully designed operations. The resulting methods were complex systems which were difficult to tune and adapt to different problems or data. A new approach to solving this problem has emerged recently: deep learning. This trend is motivated by two main factors: the increasing digitization of society, which makes it possible to record large datasets of labeled samples; and the availability of a large pool of computational resources. This work evaluates state-of-the-art deep learning methods in object detection and semantic segmentation in the field of automated surface inspection in metals. Images acquired in the industry are affected by the conditions of the environment, including noise, dust and vibrations, which are an additional challenge. Moreover, industrial inspection requires accuracy, but also robustness and speed. The selected methods are applied to different datasets of images that include the most common defects in metals and the performance is compared in terms of accuracy and speed. Results show exceptional accuracy at a fraction of the required processing time.** [1]

***Index Terms*—Surface inspection, Deep learning, Quality control, Image processing, Defect detection**

I. Introduction

In the steel industry, accurate automated quality inspection is crucial for the final quality of manufactured products. The need for high-quality products with improved performance requires accurate detection of defects. This ensures safety and overall reliability when using the manufactured products in critical infrastructures, such as buildings or transportation systems, preventing failures. Moreover, early defect detection also increases the overall manufacturing productivity, avoiding the interruption of operations and reducing production costs [2].

Automated quality control in the steel industry is widely employed in many different applications, from dimensional quality inspection [3] to temperature monitoring [4]. One area of particular importance is surface inspection [5], where the presence of defects could indicate non-critical issues that make the commercialization of the product unfeasible [6]. These surface defects include periodic scratches, or even structural problems that could lead to safety concerns, such as rolled-in material.

Defect detection and, in general, surface inspection in the steel industry is a challenging task. Inspection systems are based on cameras or other types of image sensors that must provide very good picture quality in extremely harsh environments. Sensors are generally affected by the conditions of the environment, including dust, high temperature, oil, water, vapor or vibrations [7], which not only vary with each particular installation but also with time. These varying conditions potentially affect the performance of the surface inspection procedures. Therefore, periodic maintenance procedures are required to clean the devices.

Images must be analyzed in real-time, providing early results that can be used to tune the manufacturing parameters online if required. This minimizes the number of defectively manufactured products, reducing waste and the overall production cost. However, real-time analysis and detection is difficult to achieve as steel products move very quickly at speeds ranging from 1 m/s for rails, to more than 30 m/s for steel strips during cold rolling, or even to 100 m/s for products such as wire rods [8]. Inspecting such products requires a combination of efficient algorithms and powerful hardware. Surface inspection systems must also deal with a wide variety of steel defects with different features that may change from installation to installation. Therefore, in general a complex tuning step is required before deploying an automated solution for a particular installation.

Two different approaches are used for surface inspection systems: traditional image processing methods and deep learning. In both cases, the goal is to analyze images and detect defects. Three main properties are required: accuracy, considering a metric such as precision or recall; robustness, considering the inspection methods must be applied under heavy noise conditions; and speed, as inspection needs to be performed under real-time constraints at a high frame rate. Traditional image processing methods are designed based on a sequence of specific operations on images. Low-level vision methods first transform digital images to highlight relevant features, such as edges, corners or specific intensities. Next, operations calculate regions from objects in the images and extract features such as brightness, color, size and texture.

[1]This is an extended version of the paper presented at the IEEE IAS Annual Meeting 2021 in Vancouver [1]

Finally, automatic classifiers, such as support vector machines, use the features to interpret the content of the image. The sequence of operations, or image processing pipeline, is hand-crafted by experienced computer vision engineers. It must be designed and fine-tuned for each particular problem assuming specific operating conditions. Thus, tuning complex methods is particularly difficult, with a large number of parameters affecting the performance of the system. Deep learning methods have emerged in recent years following an alternative approach known as end-to-end learning [9], in which a large dataset of sample images with annotations is used to automatically train a complex model. Deep learning methods offer great advantages over traditional methods, such as improved accuracy, albeit with some trade-offs, including the long training times and the large number of annotated samples required. The advent of deep learning methods has revolutionized the field of surface inspection in recent years, quickly replacing traditional methods with much more accurate and robust solutions.

The advent of deep learning based methods has shifted the requirements for designing high-performance systems, from feature engineering and hand-crafting complex processing pipelines to iterating through deep learning architectures and creating datasets. This new approach requires knowledge about current models, requirements, limitations and capabilities. This work evaluates the performance of different state-of-the-art deep learning methods in automated surface inspection for metals. A brief review of the advances in recent deep learning methods with applicability to surface inspection is also presented. This work is focused on two particular types of methods: object detection and semantic segmentation [10]. Both methods aim to solve the same problem: image segmentation. This is a challenging problem that involves dividing the image into regions with semantic value. In the case of surface inspection, the goal is to detect regions with anomalies or defects. One approach for solving this problem using deep learning is object detection, in which the rectangular bounding box of objects in the image are detected and classified. Another option is semantic segmentation, where pixel-level labeling is performed. This work reviews state-of-the-art methods in object detection and semantic segmentation and performs a comparative review when they are applied to surface inspection in metals. In object detection, recent architectures, such as YOLO4 and YOLOv5 [11] (You Look Only Once, YOLO, detectors version 4 and 5), claim to provide improved performance at very high-speeds. In semantic segmentation, U-Net [12] has been recently applied to pixel-level labeling in similar fields with exceptional results. In order to evaluate the feasibility of these deep learning methods for automated surface inspection, they are applied to two datasets of real surface defects that include the most common defects in metals: inclusions, crazing, patches, pitted surface, scratches and rolled in scale. The performance of object detection and semantic segmentation is compared in terms of both accuracy and speed.

This paper is organized as follows: Section II reviews the literature and recent advances in deep learning applied to surface inspection, Section III presents the selected architectures and experimental procedure to compare the performance; Section IV discusses the results obtained, and finally, Section V reports conclusions.

II. Deep Learning

Deep learning is a specific kind of machine learning mainly based on artificial neural networks. Most deep learning models extend traditional neural networks with new operations, such as convolutions, and many more layers. This provides the opportunity to train more complex models, with improved generalization capabilities, that are more accurate than previous approaches. The deep learning concept was introduced many decades ago, but only in recent years has it become the predominant machine learning method, particularly due to the spectacular results in the image classification competition in 2012 [13]. Since then, it has been applied very successfully in a wide variety of application domains. Two reasons are generally attributed to the rapid growth of this approach: the increasing digitization of society and the availability of more computational resources [14].

Very large datasets are required to train complex machine learning models. As the size of the dataset increases, the skills required to obtain an accurate model are reduced. This makes it possible to train complex models with millions of parameters using a large dataset of labeled samples. Creating this type of datasets has only been possible recently, driven by the increasing digitization of society, in which most aspects of life generate data that is recorded and labeled, such as photos, videos, or speech.

Another aspect that has contributed to the success of deep learning in recent years is the availability of cheap computational resources. The advent of graphic processing units (GPUs) that can run general purpose operations, such as convolutions, at very high speeds has dramatically changed the landscape of machine learning. Using general purpose programming on the GPUs, complex deep learning models can be trained in a fraction of the time required to train them on CPUs. Therefore, the availability of faster and cheap computational resources is one of the key enabling technologies behind deep learning.

Training deep neural networks, i.e., with a large number of layers, was difficult to solve in the past due to mathematical issues in the optimization procedure, such as the exploding gradient and the vanishing gradient. However, improvements in the gradient descent optimization algorithm and techniques such as gradient clipping, regularization, ReLU activation functions, skip connections and residual neural networks have solved or effectively diminished the issues to a great extent. The combination of these advanced techniques with large datasets and powerful hardware is behind the deep learning revolution [15] where super-human performance is achieved in very complex tasks, such as image or speech recognition.

A. Surface inspection

In [16] deep learning is applied to the detection of surface defects in rails caused by fatigue or impacts from damaged wheels. They propose a layered architecture with three main components: convolutions, activation functions (to perform mappings between linear and non-linear operations) and max-pooling to sub-sample the feature maps resulting from convolutions. Results indicate rail defects can be successfully classified with almost 92% accuracy. Moreover, increasing the number of layers in the architecture produces more accurate prediction models.

A more recent research work on the topic can be found in [17]. In this work, different architectures are compared using the dataset Neu-1800 proposed in [18]. This dataset was first used to benchmark the performance of traditional image processing methods using feature extraction, particularly using local binary pattern (LBP) features. It includes six types of surface defects commonly found in metals: rolled-in scale (Rs), patches (P), crazing (Cr), pitted surface (Ps), inclusion (In) and scratches (Sc). This same dataset is used in [17] to compare the performance of different architectures of deep learning: SSD, Faster-RCNN, YOLO-V2, YOLO-V3 and a modification of SSD. Results indicate the proposed variation of SSD is robust for metallic defect detection, achieving an average precision of 0.724.

Semantic segmentation has also been applied to surface inspection, although it is still less common than object detection. A recent work on the topic can be found in [19] where metal cracks are detected and segmented using deep learning. This work proposes a customized deep learning architecture based on U-Net. This approach is used to detect defects in titanium-coated metal surfaces. The proposed method also includes custom pre-processing and post-processing. Performance, in this case, is evaluated using the Dice score, achieving 0.916 with a precision of 0.934.

III. Experimental procedure

The proposed experimental procedure in this work to compare the performance of state-of-the-art deep learning methods is based on two datasets: Neu-1800 and Neu-900, both containing examples of defects in the metal industry. Neu-1800 is used to compare the performance of object detection methods, as it does not contain the pixel-wise annotations required by semantic segmentation methods. On the other hand, Neu-900 contains both object and pixel-wise annotations. Thus, the same dataset can be used to compare the two methods. Two main deep learning methods are compared based on the performance with the datasets: YOLOv5 and U-Net. The comparison is performed using generally accepted performance metrics. The datasets, the deep learning methods and the performance metrics are described below.

A. Datasets

1) Neu-1800: The Neu-1800 dataset, proposed by the Northeastern University (NEU) [18], contains six types of surface defects:

- Crazing (Cr): a network of fine cracks.
- Inclusion (In): metallic or non-metallic particles entrapped in the product. This material can detach from the product later, leaving a hole.
- Patches (Pa): defects resulting from oxidation or faulty pickling process.
- Pitted surface (Ps): sharp depressions in the surface of the product.
- Rolled-in scale (Rs): scale rolled into the surface of the steel strip.
- Scratches (Sc): sharp indentations in the surface of the steel strip.

All images are collected from hot-rolled steel strips using the same setting parameters and environment. The dataset includes 1800 gray-scale images, with 300 samples for each defect type. The original resolution of each image is 200 $\times$ 200 pixels. Examples of the images can be seen in Figure 1. As can be seen, the defects are very different in appearance, size and even aspect ratio. For example, scratches are very long and narrow while patches are more circular. Moreover, they include varying gray-scale levels resulting from variable illumination in industrial environments.

Images in this dataset are acquired using area-scan CCD cameras. The cameras acquire images from hot steel strips directly illuminated using LEDs. The size of the acquired gray-scale images is 1024 $\times$ 1024 pixels, but only the down-sampled images are available in the dataset. The defect detection system is composed of several cameras to cover the width of the strip with high resolution. Four cameras are considered in [18], but more cameras could be installed for increased resolutions, which could be required to detect smaller defects. The considered defect detection architecture is suitable for most systems, thus it is the ideal testing ground to evaluate the defect detection capability of advanced image processing architectures.

2) Neu900: Comparison and evaluation of deep learning models depends on the existence of annotated datasets. For object detection, a wide variety of datasets is available. However, for semantic segmentation, for which labels are required for each pixel in the image, datasets are scarce, particularly for surface inspection in metals. Recently [20] has released a dataset based on Neu-1800 that includes the pixel-wise binary maps required to train and evaluate semantic segmentation for automated surface inspection. This dataset, composed of 900 images (a subset of Neu-1800), is used in [20] to evaluate the performance of steel surface defect detection based on saliency evaluation, in this case using an improved local binary pattern (LBP) descriptor. This dataset includes annotations for both the bounding boxes and the pixel regions of the defects. For the evaluation of semantic segmentation, it provides binary maps where pixels with value 1 represent the defects, and pixels with value 0 represent the background, i.e., the area of the steel strip with no defect. In this case, the defect type is ignored, only the information about defect/no-defect is used.

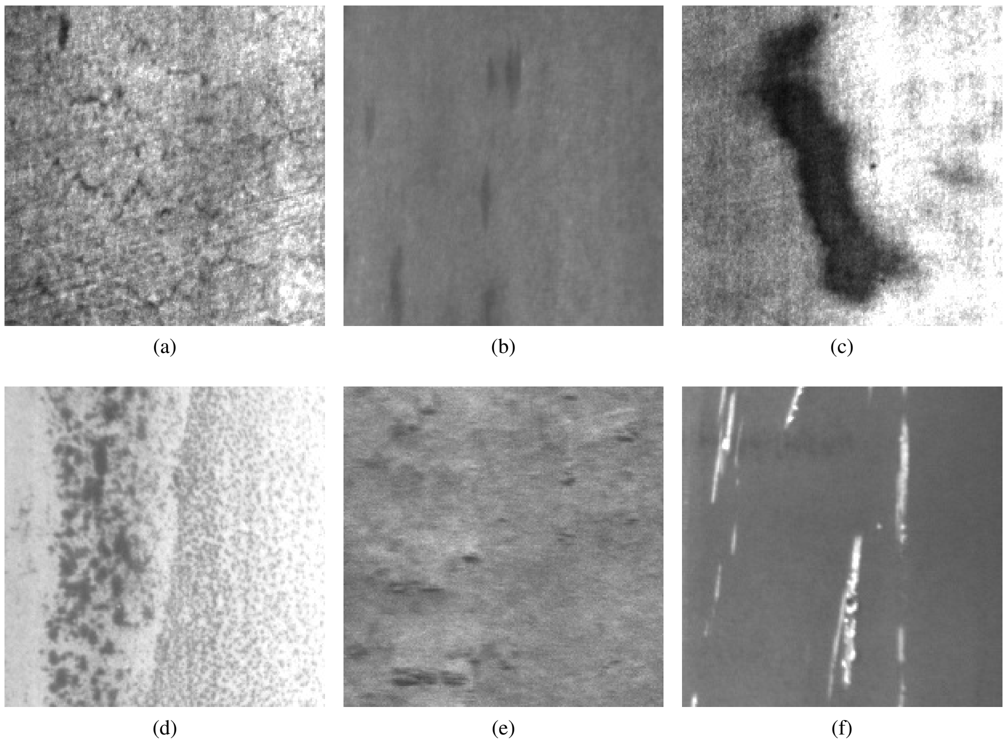

Fig. 1. Defect examples in the Neu-1800 dataset. (a) Crazing (cr). (b) Inclusion (In). (c) Patches (Pa). (d) Pitted surface (Ps). (e) Rolled-in scale (Rs). (f) Scratches (Sc).

B. Deep learning architectures

The popularity of deep learning in recent years has given rise to different deep learning architectures. The most successful are convolutional neural networks (CNNs), longshort term memory (LSTM), encoder-decoders, and generative adversarial networks (GANs). Of these, CNNs are the most widely used for image processing.

Convolution is a mathematical operation that generates one signal from two others. It is considered the most fundamental operation in digital signal processing. An input signal is usually convolved with another signal called the kernel or filter. The resulting signal highlights or diminishes an aspect of the input signal based on the selected kernel. This can be used to highlight edges or to filter noise. A single operation can be customized for a wide variety of requirements.

A convolutional neural network is mainly composed of three operations repeated in a number of consecutive layers: convolution, activation and pooling. The convolution layer performs the convolution of the input signal or image with different filters at the same time. The goal is the extraction of features. Contrary to traditional image processing, the filter coefficients are not hand-crafted, they are automatically calculated by the training algorithm using the annotated samples. The convolution is a linear operation. Thus, the result is transformed using a non-linear function, such as ReLU, to model more complex procedures. Pooling layers then replace neighboring data points using statistical information, such as the mean or the maximum. Modern CNN architectures combine a variable number of these layers stacked into a single model. For example, AlexNet, proposed in 2012, uses 8 layers while EfficientNet-B7, proposed in 2019, uses 813 layers. The first layers extract basic features, such as edges or corners, while the last layers recognize objects, shapes or positions. Modern architectures also use a combination of other operations to increase performance and accuracy, such as normalization.

Object detection networks are a particular type of CNN that aim to recognize instances of a predefined set of object types [21]. They also provide the position of the object in the image using a bounding box. Two paradigms have evolved for object detection: two-stage and single-stage object detectors. The detectors are usually composed of two parts: a backbone network trained on a large dataset such as ImageNet, and a head that predicts classes. In two-stage object detectors, the head is divided into two sub-tasks: a region proposal network that proposes regions of interest, and a classification network. Examples of this type of network are R-CNN and derivatives. Other alternative networks adopt a more unified framework to detect the objects, such as SSD and YOLO. These models are referred to as single-stage object detectors. In this case, instead of proposing regions of varying sizes and classifying them, a regression is directly performed on the bounding boxes. More recent single-stage object detectors also include additional layers between the backbone and the head, the so-called neck. The layers in the neck can be used to detect the same object with different sizes and scales.

YOLOv5 is a state-of-the-art single-stage object detector. It is considered an improved version of the YOLO detector (You Only Look Once) that has been in development since 2016. This network has been proven to be very accurate for a wide variety of datasets with real-time performance [11]. Like most single-stage object detectors nowadays, it has three main parts:

- Backbone. In this case, the CSPNet network is used for feature extraction.
- Neck. In this part, the PANet is used to calculate feature pyramids to deal with objects of different sizes.
- Head. This part deals with the final detection, including the calculation of the resulting vectors with class probabilities, confidence scores, and bounding boxes.

YOLOv5 includes improvements in data augmentation and auto-tuning of bounding box anchors rather than using fixed anchors as previous versions did. However, the main contribution is the integration of new ideas already proven in other different networks.

Semantic segmentation is different from object detection. In both cases, the problem is the same: image segmentation. However, semantic segmentation provides pixel-wise labeling rather than bounding boxes. Therefore, it provides more information about the location of the objects. On the other hand, labeling datasets is much more time-consuming because every pixel in the dataset of images needs to be assigned to a class.

U-Net is the predominant architecture used for semantic segmentation [12]. Initially designed for biomedical applications, it has proven to be very effective in other fields. U-Net is a fully convolutional network, i.e., it includes only convolutional layers. This makes it possible to take an input image of arbitrary size and producing a segmented image of the same size. The resulting image contains the label for each pixel. The U-Net architecture also follows an encoder-decoder model. The first part, the encoder, is a contracting path to capture context using convolutions. The second part, the decoder, is an expanding path in order to obtain precise localization using deconvolutions. Features from the encoder are used for the decoder. U-Net is designed to use training data very efficiently in order to learn from very few annotated images. Moreover, it admits a variable number of input channels, for example, only gray images, R, G and B channels, or even including more such as infrared. Thus, it has been used with multispectral images very successfully, for example with satellite images [22].

C. Performance metrics

In order to evaluate the accuracy of surface inspection methods, image segmentation metrics are used [23]. Most of them are based on the following notation:

1) TP: number of True Positive detections; the number of correctly detected elements, that is, hits.
2) FP: number of False Positive detections; the number of erroneously detected elements, that is, false alarms.
3) FN: number of False Negative detections; the number of undetected elements, that is, missed detections.

The number of True Negative detections (TN) is not used, as there could be a very high number of non-relevant elements in the image.

The performance in image segmentation is mostly based on two metrics: Precision (P) and Recall (R), given by (1) and (2).

$$P = \frac{TP}{TP + FP} = \frac{TP}{\text{Number of detections}} \tag{1}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{\text{Number of elements}} \tag{2}$$

Precision is the percentage of correct detections from the total number of detections. Recall is the percentage of correct detections among the number of elements. Precision measures the over-segmentation success and recall the under-segmentation success. Some methods have been proposed to combine these metrics into a single one [24], such as the F-score (F_1) (the harmonic mean of precision and recall) defined as (3). The Jaccard Index is another alternative, defined as (4).

$$F_1 = \frac{2PR}{P + R} \tag{3}$$

$$J = \frac{TP}{TP + FP + FN} \tag{4}$$

In the case of semantic segmentation, the calculation of precision and recall is straightforward. A pixel is correctly classified or not, there is no ambiguity. Thus, each pixel is counted as a TP, FP or FN. However, in the case of object detectors the procedure is not that easy.

The results of object detectors are bounding boxes of detected objects that can partially intersect with the real objects. Thus, counting them as TP depends on the intersection threshold considered. The most common method nowadays is to calculate the Intersection over the Union metric (IoU) for each detected object. IoU is defined as the ratio between the overlapping area between the detected object (D) and the real object (R), with respect to the area of the union between them, defined as (5). Only when the IoU is greater than a specific threshold (for example 0.5) is the detected object considered a true positive detection. Otherwise, it is considered a false positive.

$$IoU = \frac{|D \cap R|}{|D \cup R|} \tag{5}$$

Another popular metric for object detectors is the $Dice$ coefficient, defined as (6).

$$Dice = \frac{2|D \cap R|}{|D| + |R|} \tag{6}$$

When the $Dice$ coefficient is applied to semantic segmentation with two classes (background and foreground), the resulting metric is the same as the F-score (F_1), as can be seen in (7).

$$Dice = \frac{2TP}{(TP + FP) + (TP + FN)} = F_1 \tag{7}$$

The $Dice$ is related to IoU, as can be seen in (8). The two metrics are positively correlated, if one metric indicates a model is better than other model then the other metric will indicate the same. In general, they are considered functional equivalent, but there are differences when considering the average score over a set of inferences.

$$Dice = \frac{2IoU}{IoU + 1} \tag{8}$$

In object detection, the prediction results include a bounding box of the detected object and a confidence value that indicates which class the object belongs to in the range [0, 1], with 1 indicating maximum confidence. When only the predictions with maximum confidence are considered, the resulting precision will be high and the recall will be low. As lower confidence levels are considered, the resulting precision will decrease and the recall will increase. For varying levels of confidence, pairs of precision versus recall are obtained. This creates the precision versus recall curve, which is an accepted indicator of the accuracy of the detector. When the area under the curve is high, it indicates both high precision and recall, thus, an accurate object detector. The area under the curve, called average precision (AP), is another common metric used to assess the performance of object detectors.

When object detectors are applied to a problem with multiple object classes, for example multiple defect classes, the AP for each class are averaged. The resulting metric is known as the mean average precision (mAP), which is defined as (9), where AP_i is the average precision in the ith class and N is the number of classes.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{9}$$

The AP is always calculated for a particular threshold of IoU, which is necessary in order to calculate precision and recall. Thus, different values of AP can be obtained for different IoU thresholds. In the literature, the most common value is 0.5. This is also referred to as AP_{50}. However, in some cases the AP is averaged for different IoU thresholds. For example, the AP@50:5:95 is the average of AP for IoU thresholds from 50% (0.5) to 95% (0.95) with steps of 5% (0.05).

IV. Results and discussion

The architectures YOLOv5 and U-Net are applied to the datasets Neu-1800 and Neu-900. First, YOLOv5 is applied to the Neu-1800 dataset and the performance is compared with other deep learning architectures. Then, they are compared with traditional image processing methods using feature engineering. Both YOLOv5 and U-Net are applied to the Neu-900 dataset, as it contains labels for objects and pixels.

All the experiments are performed on a computer with an Intel Core i7 9700K CPU with 64 GB of RAM. The computer also has a GeForce RTX 2080 Ti Turbo GPU with 11 GB of RAM. The GPU is an essential tool to speed up the training process. Once the network is trained, it can be deployed in different hardware architectures with no GPU. Training on the CPU is not feasible due to the required time, but processing images may be an option depending on the requirements. The YOLOv5 network was implemented using PyTorch, while the U-Net network was implemented using Matlab. Both frameworks make efficient use of the GPU during training.

A deep learning architecture includes a large number of parameters that are learned during training using the labeled samples. In addition, the network has hyperparameters, a set of values that configure different options, such as the filter size or the number of layers. Therefore, before using a deep learning method it is necessary to tune the hyperparameters for the considered problem. There are two hyperparameter tuning methods: manual and automated. In the latter, a costly computational machine learning method is used to automatically select the hyperparameters. In this work, hyperparameters are tuned manually. This method works very well when a suitable starting point is used, which can be determined based on previous works in similar application fields. Then, variations in the learning rate, optimization method and other hyperparameters are tested for determining the best configuration.

The resulting best hyperparameters for the YOLOv5 network after the hyperparameter tuning procedure are a learning rate of 0.001, the solver Adam, 500 epochs and the small model. The YOLOv5 network includes different models that are expected to increase accuracy due to the increased complexity. In this work, the small model performed better than other more complex models (medium, large and extra-large) for the considered configurations. Before training, the Neu-1800 dataset is split into two groups: 1350 images for training and 450 for testing. Data augmentation is applied during training including clipping, rotation, flip, hue, saturation, exposure and changes in aspect ratio. Moreover, mosaic data enhancement is also applied. These augmentation techniques are used to artificially increase picture data using image processing, preventing the model from over-fitting. Training with this configuration took 49 minutes.

Figure 2 shows the precision versus recall curves for the Neu-1800 dataset with YOLOv5. These curves indicate the performance of the object detector for each defect type. Precision and recall are calculated using an IoU threshold of 0.5, the most common value in the literature.

The calculated average precision for each defect type is shown in Table I. The table shows a comparison between the performance of YOLOv5 and other networks: SSD, F-RCNN (a variant of RCNN netwok), YOLOv2, YOLOv3 and a modified version of SSD. The average precision for these architectures is obtained from [17], where the same dataset is evaluated. As can be seen, the YOLOv5 network outperforms all the other networks by a large margin. Not only does it provide the best mAP, it also provides the

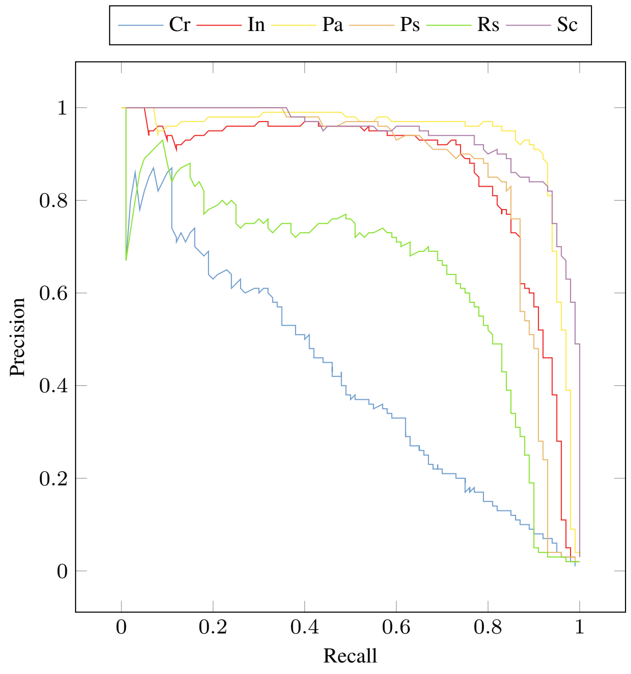

Fig. 2. Precision versus recall curves for each defect type using YOLOv5. The area under each curve is the average precision of the defect type.

TABLE I
COMPARISON OF AVERAGE PRECISION (AP) ON NEU-1800 DATASET WITH DIFFERENT ARCHITECTURES.

Defect type	Average Precision (AP) SSD	F-RCNN	YOLOv2	YOLOv3	SSD-M	YOLOv5
Cr	0.411	0.374	0.211	0.221	0.417	0.424
In	0.796	0.794	0.592	0.580	0.763	0.864
Pa	0.839	0.853	0.774	0.772	0.863	0.939
Ps	0.839	0.815	0.454	0.239	0.851	0.862
Rs	0.621	0.545	0.246	0.335	0.581	0.655
Sc	0.836	0.882	0.739	0.570	0.856	0.940
mAP	0.724	0.711	0.503	0.453	0.724	0.781

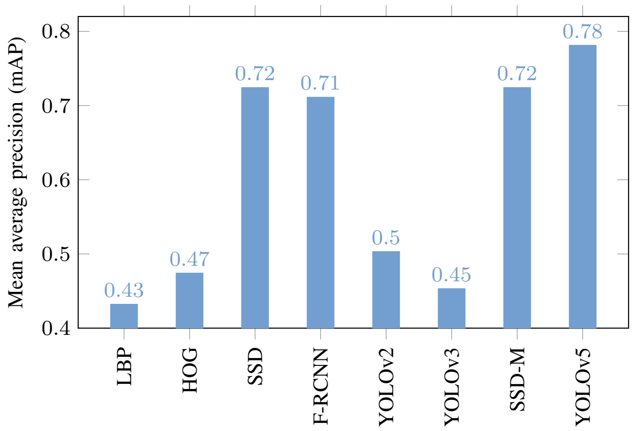

Fig. 3. Mean average precision on Neu-1800 dataset with different methods.

best average precision for every single defect type. YOLOv5 represents a major improvement with respect to the previous versions, YOLOv2 and YOLOv3. The difference lies largely in the definition of anchor boxes, the heights and widths of the bounding boxes where the objects are detected. Previous versions of YOLO used a fixed configuration prior to the training procedure. This approach did not provide good results when the objects had different scales and sizes. This is the particular case for defects, where scratches tend to be large and elongated while others have a completely different aspect ratio. YOLOv5 automatically tunes the bounding box anchors during training. The difference in performance is very noticeable, achieving state-of-the-art results on this dataset. To the best of our knowledge, this is currently the highest accuracy on the Neu-1800 dataset of any published work. Slight variations in these results could also be caused by the hyperparameters optimization of the models. However, all models are used in the best configuration based on a similar hyperparameter tuning. Thus, it is reasonable to compare the performance of these methods.

Figure 3 shows a comparison of the obtained mean average precision using different object detection methods on the Neu-1800 dataset. Two traditional methods are considered: LBP, in which features are extracted using local binary patterns and classified using support vector machines; and HOG, in which histograms of oriented gradients are calculated and also classified using support vector machines. The performance of these methods is much lower than the performance of deep learning methods, particularly of a modern detector such as YOLOv5.

Figure 4 shows some examples of defects, the ground truth and the corresponding prediction using YOLOv5. The results indicate a very high level of accuracy, with predictions very close to the ground truth in all cases. The ground truth is manually annotated by experts, however there is always some uncertainty and ambiguity in the annotation. Therefore, it is almost impossible to reach a perfect detection. YOLOv5 and other recent object detectors are said to have reached super-human performance, not just because of the high level of accuracy, but because the results are better than those a skilled technician would obtain. In this dataset, it would be extremely difficult for a technician to annotate the defects as accurately as YOLOv5; some mistakes attributed to YOLOv5 are even debatable. For example, Figure 4c shows an image with examples of patches. The ground truth has 3 annotated defects, but YOLOv5 is only predicting 2 because they are connected. This is an example where the detector is providing a result that is probably more accurate than the ground truth. On the other hand, Figure 4f shows some large and narrow scratches that go undetected, which indicates there is still margin for improvement in the network.

The next experiment evaluated both YOLOv5 and U-Net on the Neu-900 dataset. This dataset contains both labels of defects and binary masks with pixel-wise classification. In Neu-900 there is only one class: defect. This makes the dataset easier to detect and classify than Neu-1800 with six types of defects. When Neu-900 is processed with YOLOv5, the

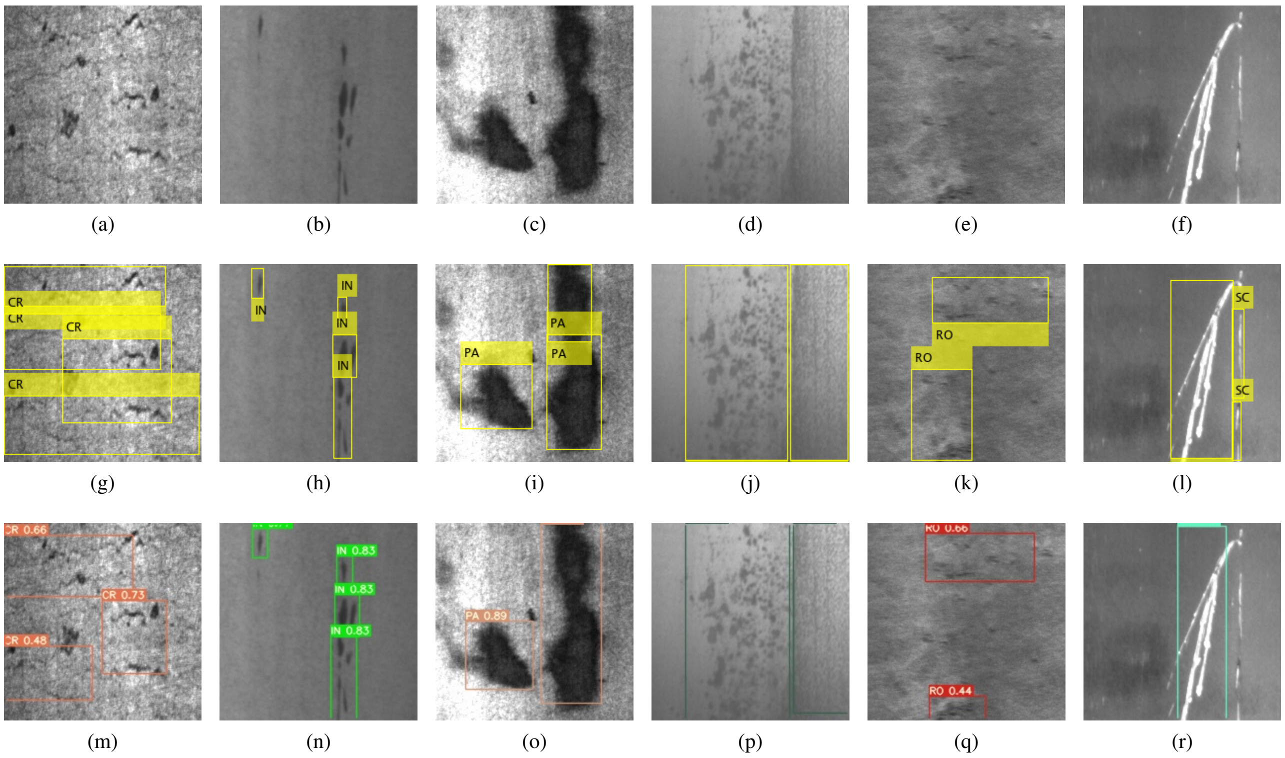

Fig. 4. Defects and detections using YOLOv5. (a)-(f) Images of defect types: Crazing (Cr), Inclusion (In), Patches (Pa), Pitted surface (Ps), Rolled-in scale (Rs) and Scratches (Sc); (g)-(l) Ground truth of the corresponding images; (m)-(r) Predictions of the corresponding images.

average precision (which in this case is the same as mAP as there is only one class) increases up to 0.893. Figure 5 shows the precision versus recall curve. This indicates that, as expected, the performance improves when the number of defect types is reduced.

Similar to YOLOv5, the hyperparameters for U-Net when using Neu-900 need to be tuned. The best configuration obtained after a variation of the most relevant parameters are: a learning rate of 0.0005, the solver Adam, 60 epochs, 3 depth levels and 32 filters (doubled in each layer). Because the number of pixels in the defect class (around 1 million) is much lower than the number of pixels in the background (almost 8 million) the classes are balanced using the median frequency weighting. Training U-Net with Neu-900 took 8 minutes. In this case, the training set is composed of 675 images; the rest is used for testing. The training progress and the evolution of the loss function during training suggest a reduced number of epochs could also be used, reducing the training time by half with similar accuracy. Training YOLOv5 with same dataset took 26 minutes as it required 600 epochs, many more than U-Net.

The average precision is a metric that cannot be used in semantic segmentation, as it requires a confidence level for each classification. On the other hand, when using semantic segmentation the results can be analyzed using a confusion matrix. Table II shows the confusion matrix for the U-Net method when applied to the Neu-900 dataset. In the field of classification, the confusion matrix is a tool commonly used to

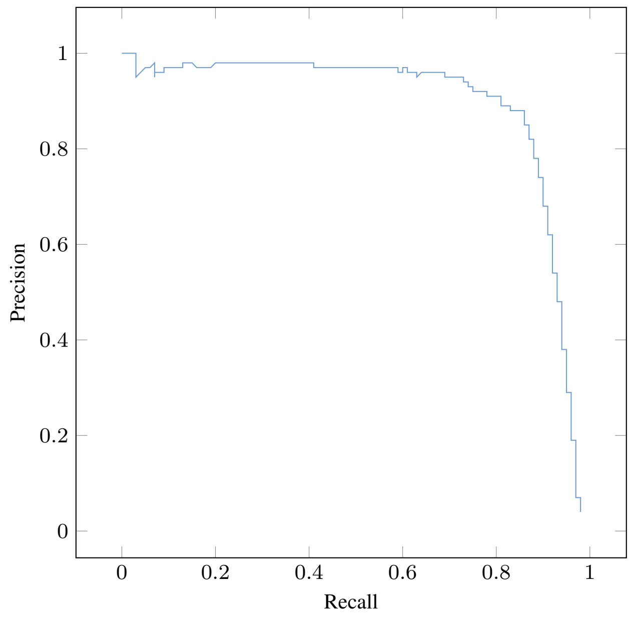

Fig. 5. Precision versus recall curve in the Neu-900 using YOLOv5. The area under the curve is the average precision (0.893).

visualize the performance of the method. The confusion matrix is a table that shows the performance of the classification method including information about TP, FP and FN. The table shows the real class of the objects and the predicted

TABLE II
CONFUSION MATRIX FOR THE NEU-900 DATASET USING U-NET.

		Real		
		Background	Defect	Precision
Prediction	Background	7 624 527	131 117	0.983
	Defect	279 718	964 638	0.775
	Recall	0.964	0.880	0.954

class. Thus, it is easy to determine whether the classifier is confusing two classes, hence the name.

Table II shows that 77.5% of the detected defects are real defects (precision). On the other hand, only 88.0% of the real defects are detected (recall). The global accuracy considering both defect pixels and background is 95.4%. Thus, the segmentation can be considered extremely accurate.

Figure 6 shows images that include defects from the Neu-900 dataset, the ground truth (considering individual defects and binary masks) and the corresponding predictions using YOLOv5 and U-Net. The results show that both YOLOv5 and U-Net provide excellent performance. In all cases, the defects are detected, with bounding boxes in the case of YOLOv5 and with binary regions in the case of U-Net.

As YOLOv5 and U-Net provide different kinds of results, it is very difficult to qualitatively compare them, as both techniques provide different but exceptionally good results. Figure 7 shows some common metrics. In the case of YOLOv5, they are calculated based on objects while in U-Net they are based on pixels. This figure shows the precision (P), the recall (R) and the F-score (F_1). All of these metrics are calculated from the TP, FP and FN. In the case of YOLOv5, a true positive detection (TP) is only considered when the intersection of the predicted bounding box and the corresponding labeled bounding box in the ground truth with respect to the union (IoU) is greater than 0.5. In the case of U-Net, a true positive detection (TP) occurs when a pixel is predicted as a defect and the same pixel is labeled as a defect in the ground truth. In both cases, the performance metrics are very high, close to one, indicating very good performance. Thus, it cannot be stated that one method is better than the other in terms of performance. U-Net has one advantage: it provides more detailed information about the location of the defects. On the other hand, annotating the dataset is much more complex and time-consuming, as each pixel needs to be assigned to a class rather than just drawing a bounding box around the defect.

YOLOv5 and U-Net can also be compared in terms of speed, by considering the required image processing time for prediction once the network is trained. Training is the time required to find the best possible parameters for the network. However, the most important aspect is the prediction time, the time required for image processing using the trained network. Figure 8 shows a comparison of the required image processing time with different deep learning architectures. YOLOv5 is faster than the other object detectors. However, the image processing time in U-Net is even lower, requiring less than 5 ms. This time is calculated for images in the Neu-900 dataset, i.e., low-resolution images (200$\times$200). However, these results indicate that these methods could be used with real-time performance in a wide variety of applications.

Figure 9 shows a comparison of the required image processing time of YOLOv5, U-Net and traditional image processing methods. There is a huge difference. YOLOv5 and U-Net can process images more than 50 times faster. This difference is also increased because deep learning methods can run on the GPU, while traditional methods are only designed to run on the CPU.

Figure 10 shows a comparison between the time required to process the image on the CPU and on the GPU. As can be seen, the GPU is 4 times faster for prediction in the case of YOLOv5. However, 28 ms per image could ideally provide a throughput of 35 frames per second, a reasonable processing speed for some applications. In the case of U-Net, the difference is much higher. The GPU is 14 times faster. U-Net is a purely convolutional neural network that can be run very efficiently on the GPU. Image processing using neural networks is largely an embarrassingly parallel problem. Thus, the current trend to increase the parallel performance of both CPUs and GPUs will clearly decrease the image processing time in the future even further.

Test are performed using the 200 $\times$ 200 low-resolution images available in the dataset. However, modern vision sensors can acquire images at much higher resolution. For example, The most common GigE cameras used in industrial environments provide 1000 $\times$ 1000 images at 100 fps. Tests performed using this type of images indicates that YOLOv5 could provide the required throughput to process these high-resolution images at 100 fps when using the GPU. Part of the network does not depend on image size but on the configured grid. Thus, keeping the same grid does not increase the processing times for that part of the network. On the other hand, the processing time with U-Net quickly increases with high-resolution images. In the case of 1000 $\times$ 1000 images it requires 24.17 ms to provide the prediction, limiting the frame rate at 40 images per second. This assumes a batch size of 1, i.e., a single image is processed at each time, emulating a real-time systems.

V. CONCLUSIONS

Automated surface inspection is of utmost importance in the metal industry. Advances in this field directly increase productivity and product quality while reducing production costs. This work evaluates a new line of image processing methods based on deep learning that can be applied to this problem. The goal is to evaluate the feasibility of these methods for defect inspection in terms of accuracy and speed.

The results obtained with YOLOv5 and U-Net, two state-of-the-art methods in object detection and semantic segmentation, are exceptional. YOLOv5 is applied to a large dataset of images that include the most common defects in metals. The results indicate that, not only is YOLOv5 the most accurate

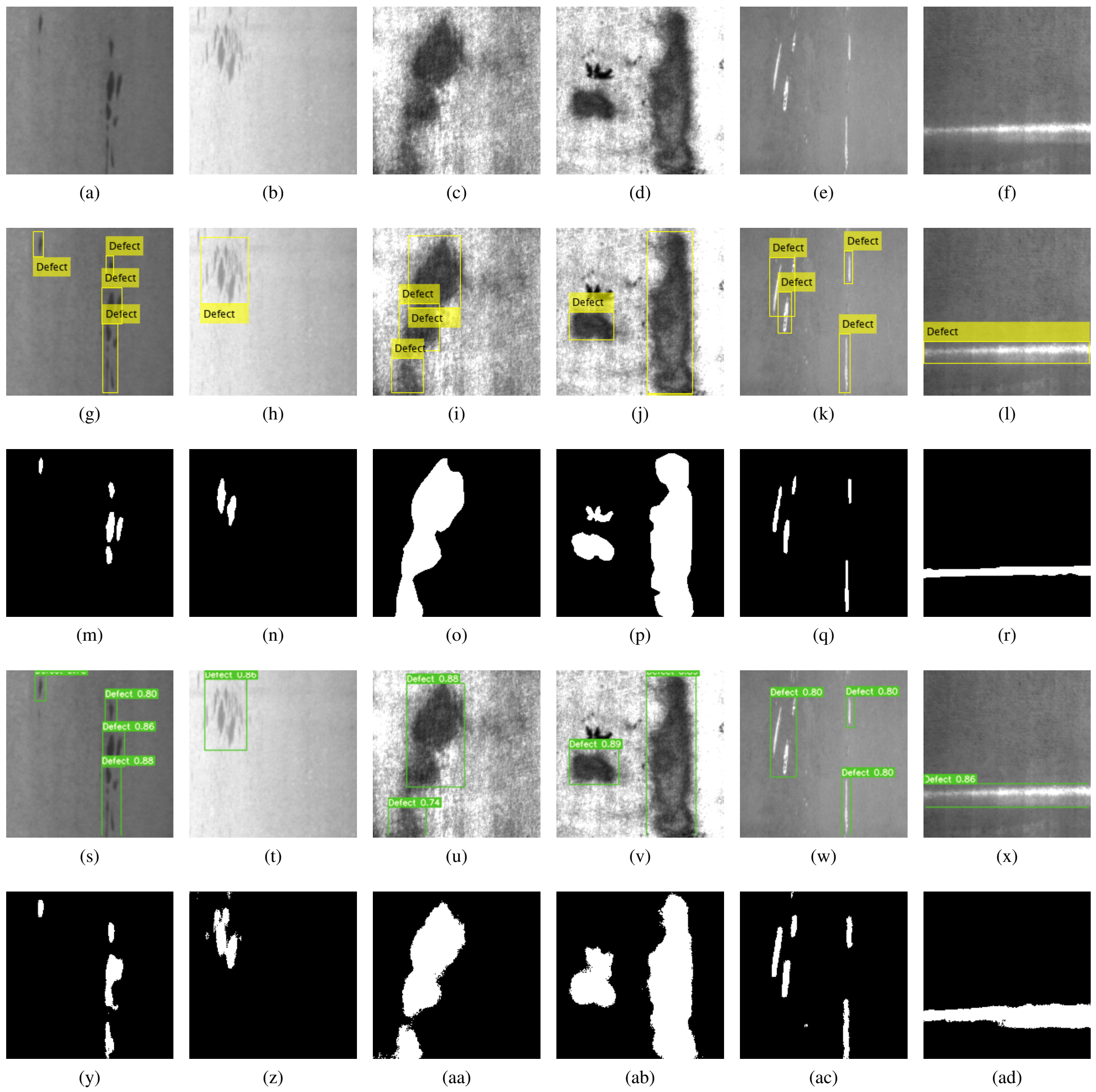

Fig. 6. Examples of detections using YOLOv5 and U-Net. (a)-(f) Images of defects; (g)-(l) Ground truth for object detection for the corresponding images; (m)-(r) Ground truth for semantic segmentation for the corresponding images. (s)-(x) Detections using YOLOv5 for the corresponding images; (y)-(ad) Semantic segmentation using U-Net for the corresponding images.

object detector compared with previous methods, it is also the fastest. When both YOLOv5 and U-Net are applied to a compatible dataset, they both demonstrate an extremely good ability to distinguish defects. U-Net can even generate binary masks with the precise location of the defects.

Deep learning methods are a revolution in the field of surface inspection. The accuracy of these methods is far superior to that of the traditional methods at a significantly reduced computational cost, providing the foundations for better and faster surface inspection systems. Real-time surface inspection systems can easily take advantage of these methods without major architectural changes. In a basic configuration, an array of cameras can be used to acquire images at high frame rates from the products are they are moved forward. Deep learning methods can be used to provide real-time feedback about possible defects or manufacturing anomalies.

Acknowledgements

This work has been partially funded by the project RTI2018-094849-B-I00 of the Spanish National Plan for Research, Development and Innovation.

References

[1] R. Usamentiaga, D. Lema, O. Pedrayes, and D. Garcia, “Automated surface defect detection in metals: a comparative review of object

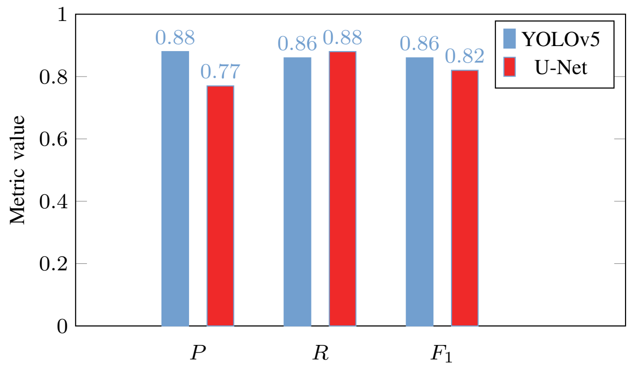

Fig. 7. Performance metrics with Neu-900 when using YOLOv5 and U-Net.

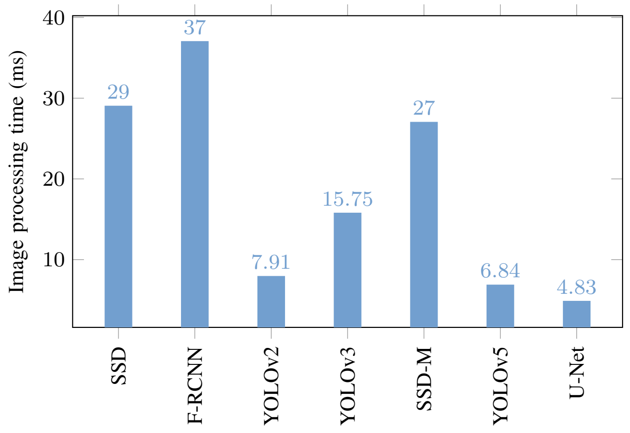

Fig. 8. Image processing time with different deep learning architectures.

detection and semantic segmentation using deep learning," in *2021 IEEE Industry Applications Society Conference*, 2021, pp. 1–8.

[2] S. L. Robinson and R. K. Miller, *Automated inspection and quality assurance*. CRC Press, 1989, vol. 16.

[3] R. Usamentiaga, D. F. Garcia, and F. J. delaCalle Herrero, "Geometric reconstruction and measurement of long steel products using 3-d sensors in real time," *IEEE Transactions on Industry Applications*, vol. 55, no. 5, pp. 5476–5486, 2019.

[4] R. Usamentiaga, D. F. Garcia, and J. M. Pérez, "High-speed temperature monitoring for steel strips using infrared line scanners," *IEEE Transactions on Industry Applications*, vol. 56, no. 3, pp. 3261–3271, 2020.

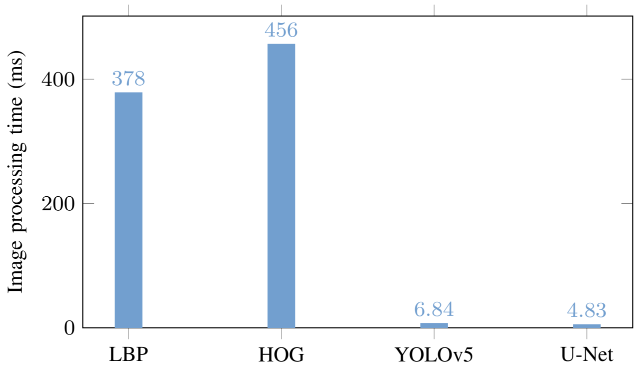

Fig. 9. Image processing time when using feature-engineering methods and deep learning methods.

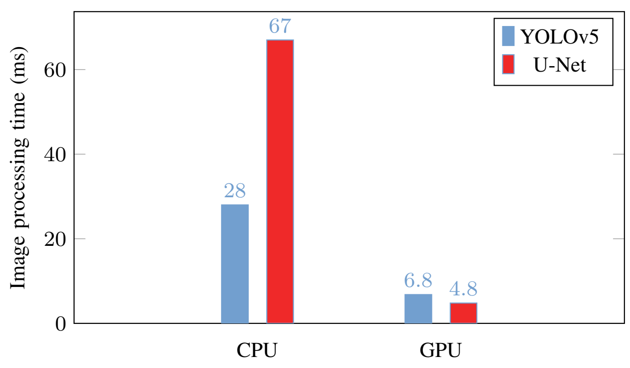

Fig. 10. Image processing on the CPU and on the GPU.

[5] N. Neogi, D. K. Mohanta, and P. K. Dutta, "Review of vision-based steel surface inspection systems," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, pp. 1–19, 2014.

[6] F. G. Bulnes, R. Usamentiaga, D. F. Garcia, and J. Molleda, "An efficient method for defect detection during the manufacturing of web materials," *Journal of Intelligent Manufacturing*, vol. 27, no. 2, pp. 431–445, 2016.

[7] R. Usamentiaga, J. Molleda, and D. F. Garcia, "Structured-light sensor using two laser stripes for 3d reconstruction without vibrations," *Sensors*, vol. 14, no. 11, pp. 20 041–20 063, 2014.

[8] H. Jia, Y. L. Murphey, J. Shi, and T.-S. Chang, "An intelligent real-time vision system for surface defect detection," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 3. IEEE, 2004, pp. 239–242.

[9] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Science and Information Conference*. Springer, 2019, pp. 128–144.

[10] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *arXiv preprint arXiv:2001.05566*, 2020.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[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*. Springer, 2015, pp. 234–241.

[13] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[14] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[15] T. J. Sejnowski, *The deep learning revolution*. Mit Press, 2018.

[16] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," in *2016 International joint conference on neural networks (IJCNN)*. IEEE, 2016, pp. 2584–2589.

[17] X. Lv, F. Duan, J.-j. Jiang, X. Fu, and L. Gan, "Deep metallic surface defect detection: The new benchmark and detection network," *Sensors*, vol. 20, no. 6, p. 1562, 2020.

[18] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858–864, 2013.

[19] Y. Aslam, N. Santhi, N. Ramasamy, and K. Ramar, "Localization and segmentation of metal cracks using deep learning," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2020.

[20] G. Song, K. Song, and Y. Yan, "Saliency detection for strip steel surface defects using multiple constraints and improved texture features," *Optics and Lasers in Engineering*, vol. 128, p. 106000, 2020.

[21] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[22] M. Freudenberg, N. Nölke, A. Agostini, K. Urban, F. Wörgötter, and C. Kleinn, "Large scale palm tree detection in high resolution satellite images using u-net," *Remote Sensing*, vol. 11, no. 3, p. 312, 2019.

[23] R. Padilla, S. L. Netto, and E. A. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020, pp. 237–242.

[24] R. Usamentiaga, D. F. García, C. López, and D. González, "A method for assessment of segmentation success considering uncertainty in the edge positions," *EURASIP Journal on Advances in Signal Processing*, vol. 2006, pp. 1–12, 2006.