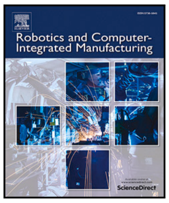
Robotics and Computer–Integrated Manufacturing 80 (2023) 102470

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



Full length article

Deep learning based online metallic surface defect detection method for wire and arc additive manufacturing

Wenhao Li [a], Haiou Zhang [a], Guilan Wang [b], Gang Xiong [c,d], Meihua Zhao [e,f], Guokuan Li [g], Runsheng Li [a,*]

[a] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, China
[b] School of Materials Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, China
[c] Beijing Engineering Research Center of Intelligent Systems and Technology, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[d] Guangdong Engineering Research Center of 3D Printing and Intelligent Manufacturing, Cloud Computing Center, Chinese Academy of Sciences, Dongggguan 523808, China
[e] State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[f] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[g] Wuhan National Laboratory for Optoelectronics, Wuhan, Hubei, 430074, China

ARTICLE INFO

Keywords:
Wire and arc additive manufacturing
Defect detection
Online
Deep learning

ABSTRACT

Wire and arc additive manufacturing (WAAM) is an emerging manufacturing technology that is widely used in different manufacturing industries. To achieve fully automated production, WAAM requires a dependable, efficient, and automatic defect detection system. Although machine learning is dominant in the object detection domain, classic algorithms have defect detection difficulty in WAAM due to complex defect types and noisy detection environments. This paper presents a deep learning-based novel automatic defect detection solution, you only look once (YOLO)-attention, based on YOLOv4, which achieves both fast and accurate defect detection for WAAM. YOLO-attention makes improvements on three existing object detection models: the channel-wise attention mechanism, multiple spatial pyramid pooling, and exponential moving average. The evaluation on the WAAM defect dataset shows that our model obtains a 94.5 mean average precision (mAP) with at least 42 frames per second. This method has been applied to additive manufacturing of single-pass, multi-pass deposition and parts. It demonstrates its feasibility in practical industrial applications and has potential as a vision-based methodology that can be implemented in real-time defect detection systems.

1. Introduction

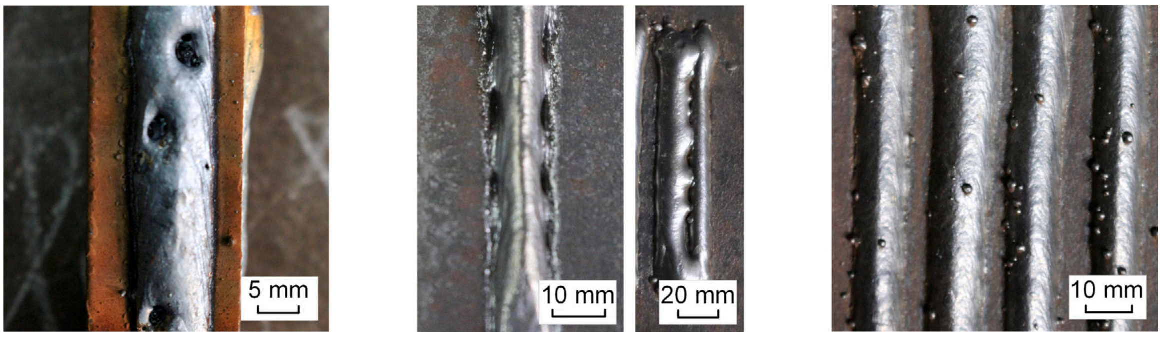
Wire and arc additive manufacturing (WAAM) [1,2] is a wire-feed directed energy deposition technology. It adopts an electrical arc as its heat source for fusion to melt the metal feedstock and accumulate it to form a rudimentary part layer by layer [3–5]. Compared with other techniques, WAAM has reasonable precision and much higher efficiency because its deposition rate is on the order of 50–130 g/min, while a rate of 2–10 g/min is achieved for laser-based and electron-based metal additive manufacturing technologies [6–8]. In addition, WAAM specializes in making large and high-strength parts. Some advanced manufacturing industries have adopted WAAM to fabricate intricate metal workpieces, such as titanium alloy and aluminium alloy load-bearing frame parts for aeroplanes [9–11] and high-strength alloyed pumps for ships [12]. However, because of the extensive energy input needed and the liquid state of the melting metal material, bottleneck problems such as cracking, shrinkage porosity, and deformation occur frequently, which seriously affect the surface tolerance and forming accuracy of the final workpiece.

To achieve the integration and high performance of the whole process, this progressive industry requires a closer collaboration between manufacturing and defect detection. While automatic WAAM is prevalent, an efficient and real-time defect detection system is highly desirable but still lacking. Manual detection techniques not only are associated with dangerous conditions, such as high temperatures and specific gasses [13], but also require operators with great expertise. In addition, transferring workpieces between automated and manual production lines is time consuming. To ensure an integrated industrialized time-intensive weld defect detection system, it is important to establish an inspection algorithm to monitor weld bead processing in real time.

In summary, the formidable challenges for researchers of a real-time WAAM defect detection system are threefold. First, the slag inclusions

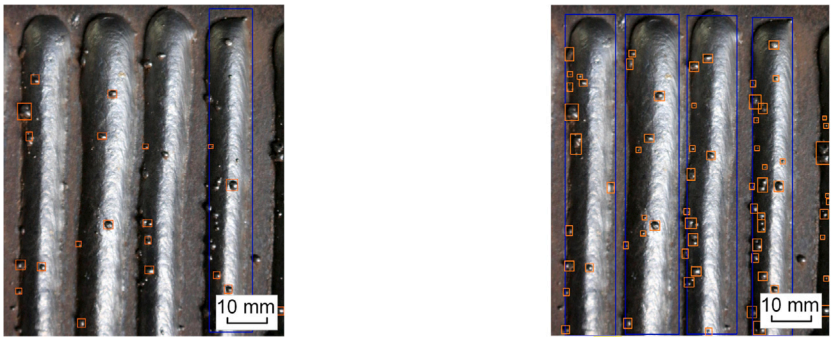
* Corresponding author.
E-mail address: lirunsheng@hust.edu.cn (R. Li).

https://doi.org/10.1016/j.rcim.2022.102470
Received 11 December 2020; Received in revised form 5 February 2022; Accepted 26 September 2022



(a) surface pores (b) grooves (c) slag inclusions

Fig. 1. Defects in wire and arc additive manufacturing. 15(a): Different types of surface pores on the weld beads (three large surface pores and one small surface pore). 15(b): Groove problems occurred at three locations. 15(c): Numerous slag inclusions on the weld bead.



(a) YOLOv4 (1 weld and 20 slag inclusions detected) (b) Ours (4 welds and 67 slag inclusions detected)

Fig. 2. Performance comparison between YOLOv4 and our method (confidence threshold 0.5).

and surface pores in the weld bead are often small and dense, which leads to difficulties in recognizing them. Second, the background of the object is deceptive because some defects do not need to be detected, such as slag inclusions in the metal base (Fig. 15(c)). Third, the three types of defects, slag inclusions, surface pores, and grooves (Fig. 1), vary in shape and size and do not have common characteristics that allow us to distinguish and locate them.

To apply the object detection model in deep learning to defect detection for WAAM, we manually produce a dataset containing four kinds of targets: welds, surface pores, grooves, and slag inclusions. An undercut is an irregular groove that is formed between the substrate and the deposited material. The lack of fusion in multi-pass deposition is the groove between the weld beads. The two defects are similar in image, so they are summarized as grooves defects in this research. The dataset we produce contains 760 images of welds taken by a CMOS camera, with a total of 8583 objects. These images contain data from our actual industrial production and harsh conditions created by manual experiments.

You only look once, version 4 (YOLOv4) [14] is a popular object detection model that achieves both accuracy and efficiency. For example, when processing images on a 15.3 TFLOP accelerator (NVIDIA GTX 1080Ti), YOLOv4 achieves an accuracy and recall rate of 43 mAP (mean average precision) on the COCO [15] dataset and a processing rate of 45 images per second (a.k.a., frames per second). This processing rate is sufficient to meet the object detection rate requirement both in academia [16] and industry [17] (30 frames per second is the minimum requirement for real-time detection generally).

YOLOv4 achieves both efficiency and accuracy through three major design choices. First, unlike the two-stage target detection model, it classifies and regresses data only once, which saves a large amount of detection time, making it fast enough to be deployed online. Second, the residual structure it uses over the backbone network enables it to extract more in-depth information with high efficiency. Third, the feature fusion of the feature pyramid network (FPN) preserves the information of small targets to improve its detection effect. However, applying YOLOv4 to WAAM defect detection tasks is impeded by one main difficulty: bare-metal YOLOv4 often overlooks minor defects (only 48.8 mAP for slag inclusions on the weld bead in our experiment, as shown in Fig. 2(a)). In our experiment, the original YOLOv4 achieves only 72.8 mAP for all objects, which is not sufficient for application in actual industrial production.

To adapt to the requirements of defect detection in WAAM, we establish our WAAM defect detection system on YOLOv4 and optimize our model specifically for the two original problems with three primary techniques—channel-wise attention, multiple spatial pyramid pooling (SPP), and exponential moving average (EMA)—where the first two techniques are used to enhance the detection performance against small targets, and the last is used to stabilize the training process and make the model converge to the optimal global solution. In addition to these techniques, we adopt some strategies to optimize and strengthen the effect of the final model, such as mosaic data augmentation, cosine decay, and transfer learning.

In summary, the main contributions of our work are as follows:

- We propose a deep learning-based object detection algorithm for defect detection in WAAM. By learning from a large number of samples, four kinds of defects can be accurately detected from input images.
- We propose YOLO-attention to improve the performance of the defect detection model, which is based on YOLOv4; YOLO-attention provides improvements in three primary techniques: channel-wise attention, multi-SPP, and EMA.
- We conduct sufficient experiments to verify the effectiveness and robustness of our proposed method. The results show that our approach achieves 94.5 mAP on our defect dataset for accurate

non-destructive online defect detection at 42 frames per second (FPS). Our method eliminates redundant slag inclusions in a complex background and identifies real defects (Fig. 2).

2. Related work

2.1. Traditional defect detection technologies

The recent advancement in artificial inspection is promising for research on improving the accuracy of detection while ensuring efficiency. Some existing commercial automatic defect detection systems are available. Atomic titanium and vanadium optical emissions have been found to be correlated with defects [18], but the type of materials directly limits this method. Everton et al. [19] detected manufactured surface pores by using laser ultrasonic testing. However, their technique cannot detect multiple defects and is too slow to be viable for in situ process inspection. In the area of traditional computer vision, such as the model-based method [20–22], spectral method [23–25], structural method [26–29], and threshold method [30–34], the approaches require stringent data acquisition conditions and precise thresholds set for different defects, which not only define application scenarios but also reduce the accuracy rate. For other technologies, such as ultrasound [35–37], infrared thermal imaging [38,39], and spectroscopy [40,41], these methods achieve reasonable results in their original fields, but the large equipment and restrictions on specific materials limit their comprehensive application in WAAM.

2.2. Defect detection using CNNs

Convolutional neural networks (CNNs) have fuelled great strides in the domain of computer vision and made significant breakthroughs in many areas, such as image classification (e.g., [42,43]) and object detection (e.g., [44,45]). In addition to doing theoretical work, researchers are trying to apply CNNs to the industrial inspection system to increase their practical value, especially in defect detection.

To obtain the locations and types of defects directly, researchers have begun to combine industrial production with object detection algorithms. Tao et al. [46] designed a novel cascaded autoencoder architecture to segment and localize defects on a metallic surface. However, this method cannot distinguish between different individuals of the same type, and the background of the objects being detected is not as complicated as in WAAM. Lin et al. [47] adopted the faster region-based convolutional neural network (Faster-RCNN) [48] for defect detection on a steel surface. Since Faster-RCNN is a two-stage model, the speed of processing images is only 5 FPS, which greatly limits its application in real-time industrial inspection. Vigneashwara et al. [49] segmented weld bead images at the pixel level by using an encoder–decoder architecture based on VGG-16. However, it cannot be applied to defect detection in WAAM because the conclusion is obtained on the premise of only one object being detected in each image.

One-stage object detection is an algorithm that classifies and regresses data only once, which is time saving at the expense of a slight amount of accuracy compared to the two-stage method. However, to perform real-time (at least 30 FPS) defect detection, the one-stage algorithm is usually the only choice. YOLO is a classic one-stage object detection algorithm, and YOLOv4 achieves state-of-the-art accuracy and efficiency. In the academic area, Huang et al. [50] applied dense connections and the SPP to the architecture of YOLOv2 [51], increasing the accuracy of the original model by 2.25%. For industrial applications, S. Yanan et al. [17] used YOLOv3 [52] to detect defects in rail surfaces and achieved a 97% recognition rate. However, these methods did not optimize YOLO for the small defects in WAAM specifically. To adapt it to the defect detection tasks of WAAM based on state-of-the-art performance, we carry out specific optimizations for minor defects on YOLOv4.

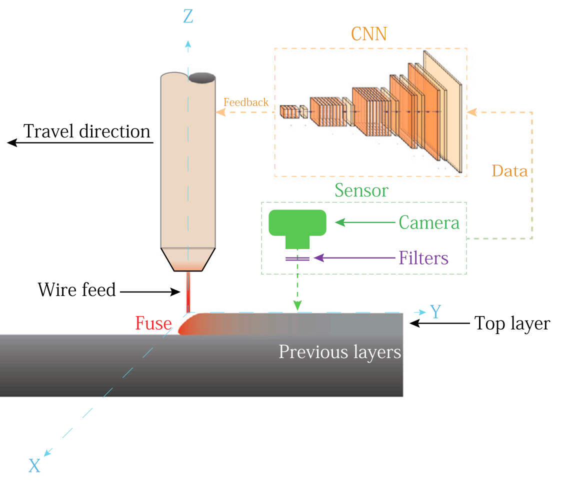


Fig. 3. Schematic diagram of the visual sensor system.

3. System overview

Fig. 3 shows the diagrammatic layout of the passive visual sensor system and the subsequent processing flow. As shown in Fig. 3, the vision sensor consists of a CMOS camera and an optical filter. The manufacturing coordinate is set as follows: the X-axis is the deposition width direction, the Y-axis is opposite to the travel direction, and the Z-axis is the deposition height direction. The MIROEX4-4096MM digital CMOS camera is competent in acquiring 8-bit RGB images at a resolution of 1920 × 1080 pixels with a sampling frequency of 60 FPS. To eliminate the radiation of arc light, in addition to limiting the distance between the sensor and the molten pool, we place an optical filter acting as a hard light barrier in front of the CMOS camera.

In our system, we define four kinds of targets to be detected: welds, surface pores, grooves, and slag inclusions. As shown in Fig. 3, the data obtained from the vision sensor are subsequently transmitted to the CNN model, which consists of the backbone network, FPN, and detection layer. The backbone network and FPN are responsible for the extraction and fusion, respectively, of the features of the input images. The detection layer synthesizes the features to output specific information and customizes the generated anchors to fit objects.

4. Method

This section describes the four primary components of our model structure: the backbone network, FPN, multiple SPPs, and detection layer. In particular, the backbone network and detection layer are the core parts of the model, while the FPN and SPPs are added to the model to enhance its ability to detect small targets. The features of the input image are extracted by the backbone network and fused in the FPN. This information is used to mesh the original image and generate anchors and adjustment parameters for each mesh to fit targets.

4.1. Backbone

A backbone network plays the role of extracting features from input images in a detection model, which directly determines the performance of the final model. DarkNet53 is the backbone network in YOLOv3, and some changes are made based on it in YOLOv4. However, in WAAM defect detection task, we find that these changes not only do not improve the effect adequately but also increase the computational burden. To improve the performance of detection, we propose DarkNet53-attention, wherein a channel-wise attention structure (Fig. 5) is added to the residual blocks based on DarkNet53. Furthermore, the activation functions and batch normalization (BN) layer

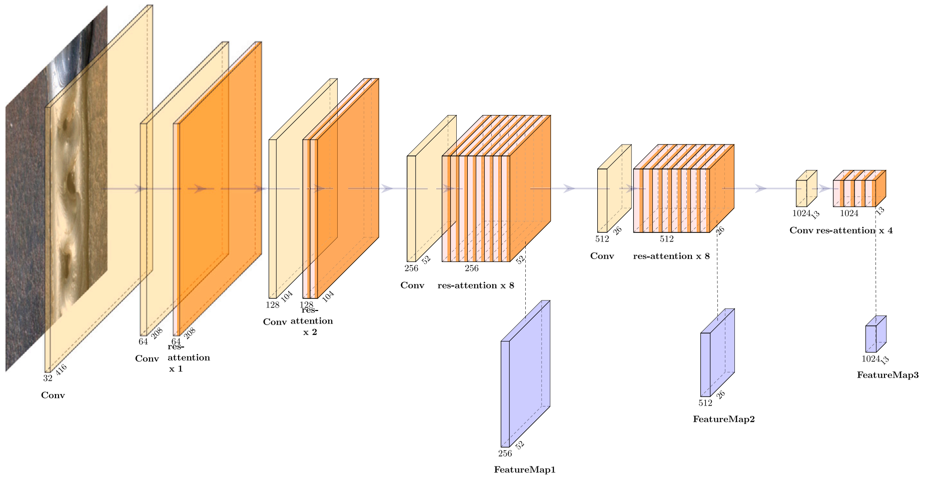


Fig. 4. The architecture of DarkNet53-attention (assuming the size of the input image is 416).

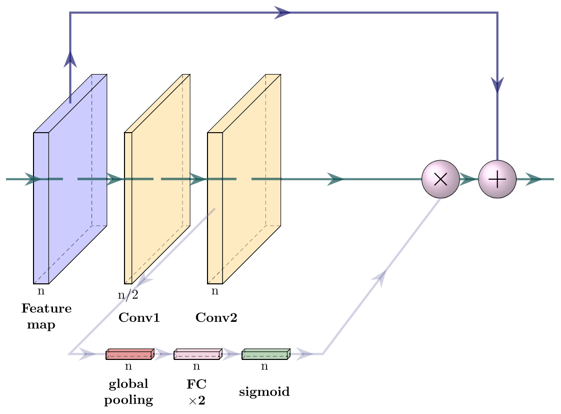


Fig. 5. A block of res-attention network.

momentum are modified. The architecture of DarkNet53-attention is shown in Fig. 4.

It can be observed that DarkNet53-attention consists of Conv layers and res-attention blocks. Each Conv layer is composed of convolution, BN, and an activation function. The res-attention block is a residual attention structure. As shown in Fig. 5, we adopt the structure of residual blocks. The architecture of a residual block creates a shortcut structure to add the original feature maps to the output. Subsequently, ignoring the activation function, BN layer, and attention branch, we obtain the relationship between the output and input as

$$y_i = \mathcal{F}(x_i, \{W_i\}) + x_i, \tag{1}$$

where $\mathcal{F}(x_i, \{W_i\})$ convolves x_i with a weight of W_i. With the shortcut structure, our CNN model is capable of extracting information of more diverse forms from the input images without reaching saturation. This is fundamental for accurately locating defects.

In the res-attention block, we assign an evolutionary weight to the different channels in the feature maps since we consider the information in each channel to be unequally valuable. In WAAM defect detection tasks, we need to emphasize the details in the image. Specifically, the attention branch contains a global pooling layer, two full connection (FC) layers, and a sigmoid activation function (Fig. 5). For the operation process, the first two change the dimension of the feature maps to $n \times 1 \times 1$, and the sigmoid function outputs n numbers between 0 and 1 as the weight of each channel, where n is the number of channels in the feature maps. In the final stage, we multiply the n output numbers by the n channels of the original feature map to emphasize the information in a particular channel. By training with a large number of small targets, the model can use self-learning to enhance the channels with small target information.

During training, we modify the input image size by a linear interpolation algorithm. Our experiments demonstrate that the input size will affect the balance between accuracy and efficiency. We illustrate this in the data augmentation Section 5.2.

We modify the BN layer and activation function. The BN layer is a technique to normalize the feature map and standardize the data distribution. There are two evolutionary parameters γ and β in the BN layer. To weaken the interference of unstable factors for a certain batch of images, we add a momentum parameter and set it to 0.03. The momentum keeps the evolving trend of the parameters to make the update curve smoother and prevent significant fluctuation due to interference. In addition, the rectified linear unit (ReLU) and Leaky ReLU activation functions cause partial information loss when the weight is less than 0, which is not suitable for us to detect small targets. Consequently, we change the activation function to the parametric ReLU (PReLU) [53]:

$$PReLU(x_i) = \begin{cases} x_i & \text{if } x > 0 \\ p_i x_i & \text{if } x \le 0 \end{cases} \tag{2}$$

As shown in Eq. (2), PReLU is an advanced form of Leaky ReLU because the coefficient of x is evolutionary when x is less than 0. In Eq. (2), p_i is a coefficient that is set separately so that each channel in the feature map retains more information on essential channels.

4.2. Feature pyramid networks

The receptive field of the network increases after multiple convolutions, which obliterates detailed information. To address this problem, we reserve multiple feature maps to carry out a series of feature fusions. Assuming the input image size is 416 × 416, after being preprocessed by the backbone network, the input corresponds to outputs of three different sizes, 52 × 52, 26 × 26, and 13 × 13 (the original size divided by 8, 16, and 32), which are named FeatureMap1, FeatureMap2, and FeatureMap3, respectively. Feature maps of different sizes contain information of different depths. These three feature maps will be subsequently introduced into the FPN for further feature fusion to enhance the necessary information.

The FPN [54] is an architecture for fusing feature maps of various sizes. Some classical object detection models, such as the single-shot detector (SSD), do not employ FPNs in their network architecture, resulting in a high efficiency but inferior detection effect for tiny targets. However, WAAM requires a satisfactory criterion for detecting extremely small defects. The FPN of YOLOv4 is a lightweight network

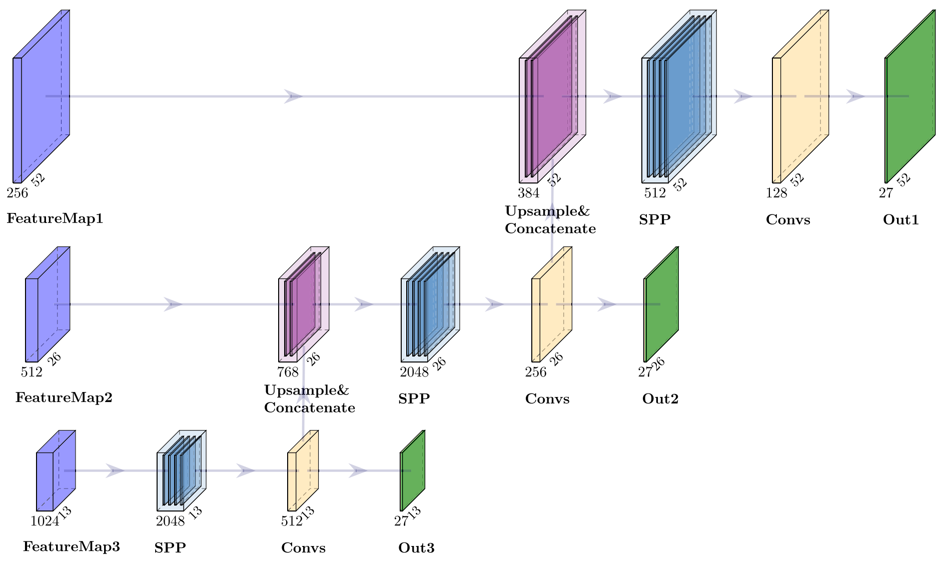


Fig. 6. The structure of the FPN.

with reasonable performance. To further improve the detection effect of small targets based on YOLOv4, we attach a multiple-SPP structure to the FPN and modify the BN layer and activation functions. The architecture adopted for the FPN is shown in Fig. 6.

The inputs of the FPN are FeatureMap1-3. Assuming the size of the backbone input image is 416 × 416, the three feature maps we obtain have sizes of 52 × 52, 26 × 26, and 13 × 13. In the FPN, FeatureMap3 will first carry out the SPP operation and a series of convolutions to acquire Out3 (13 × 13), which is equivalent to dividing the input image into 13 × 13 grids, where each grid corresponds to a value of Out3. Since multiple convolutions obliterate the details of the original image, Out3 is only appropriate for extracting information from large objects. Next, we upsample FeatureMap3 to fit the size of FeatureMap2 and concatenate them. In this way, we combine deep information with shallow information. After that, we obtain Out2 by another SPP operation and a series of convolutions. By repeating the above manipulations, the FPN exports three outputs named Out1-3. In contrast to Out1, Out3 is more appropriate for extracting details to detect small targets. In addition, we modify the BN layers in this part to add a momentum of 0.03, and the activation function adopted is still PReLU.

This kind of FPN is not the most accurate approach for fusing features. For example, in EfficientDet [55], the author uses a bi-directional FPN (BiFPN), a much more sophisticated structure than FPN, leading to a higher upper bound than our method. Correspondingly, it is time consuming and takes up many more GPU resources, so it is not capable of being used in WAAM applications. We carry out relevant experiments and demonstrate the results in Section 6.4.

4.3. Spatial pyramid pooling

To enhance the adaptability of the model to different scale features, the original YOLOv4 appends an SPP [56] structure to the FPN. However, our experiment proves that one SPP structure has a limited effect on the model, so we use multiple SPPs to add an SPP structure for each of the three branches of the FPN. The architecture of the SPP network is shown in Fig. 7. In simple terms, SPP is a combination of maxpooling operations and concatenation. In short, the three maxpooling operations are set to execute with the same stride but different sizes. Due to having the same strides and the same padding, the size of the output feature maps is equal to the original, which ensures that the outputs can be concatenated to the input.

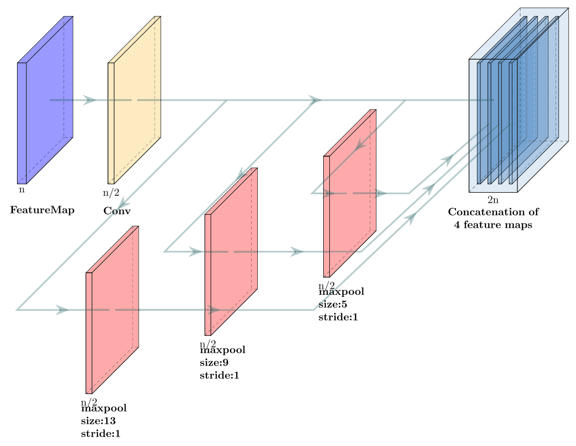


Fig. 7. Visualization of spatial pyramid pooling.

According to the original author's point of view, various sizes of maxpooling can be adopted. However, based on the results of our experiments, we conclude that the best maxpooling performance is achieved with sizes 5, 9, and 13. After that, we concatenate three outputs to the original feature map to obtain a feature map with four times the number of channels as in the input.

4.4. Detection layer

The detection layer is a section that further processes Out1-3 to generate the prediction bounding boxes. For each grid in the three feature maps, we generate three kinds of anchors of different sizes. By adjusting the centre coordinates, length, and width, the detection layer modifies the anchors to the final prediction bounding boxes.

In detail, the detection layer adjusts the number of channels in the last dimension of Out1-3 to $3 \times (4 + 1 + n)$, where 3 is the number of anchors per grid, 4 is the number of adjustment parameters of one anchor, 1 is the confidence, and n is the number of classification categories. In our experiment, n is equal to 4. Fig. 8 shows an example of the three anchors for one grid. By setting a threshold, we filter out the targets with low confidence. In addition, we also eliminate the redundant prediction boxes through the non-maximum suppression (NMS) method. With these two screenings, we eliminate all the

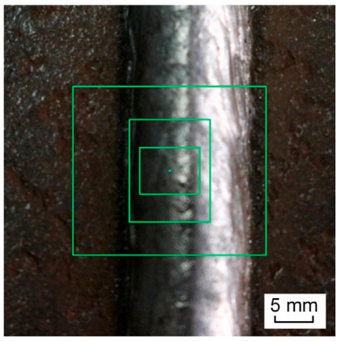


Fig. 8. Visualization of the anchors for one point.

unnecessary bounding boxes and preserve the locations and sizes of targets.

In addition to NMS, we also tested soft NMS [57]. Soft NMS is a method that can improve the detection effect when two targets overlap considerably. The results demonstrate that soft NMS does not have a positive effect on the performance. There are almost no overlapping defects in the process of WAAM, so the more computationally intensive soft NMS does not need to be applied.

5. Training

This section introduces our strategies and tricks during training. We used the EMA to optimize the weight updating curve, which significantly improves the performance of the final model with a small computational burden increase. In addition, we adopted an appropriate learning rate schedule to accelerate the training process and modify the anchor size to promote the fitting degree of the prediction bounding box with the ground truth. We also used data augmentation and transfer learning to address the problem of insufficient datasets in industrial applications. Finally, we introduced loss functions and emphasized the complete intersection over union (CIoU) function.

5.1. Exponential moving average

Based on the backpropagation principle, the GPU calculates the variations of weights according to their gradients and the loss function value of a batch of images. The optimization process deviates readily from the prospective track due to the potential interference in a batch of images, which is most evident at the initialization and termination of the training process. Furthermore, in the later stage of training, the model oscillates near the optimal solution and is unable to reach it.

To address the problems above, we decided to apply the EMA to training to relate each step of weight optimization to the previous steps. Assuming that the weights of the convolutional layer w are denoted as w_t at time t, θ_t is the value of w at time t. If the EMA is not used, $w_t = \theta_t$. If the EMA is adopted, the formula for w_t is updated to:

$$w_t = \beta \cdot w_{t-1} + (1 - \beta)\theta_t \tag{3}$$

where $\beta \in [0, 1)$. $\beta = 0$ means there is no use of EMA. Eq. (3) shows that the value of w at time t is approximately equal to the average of θ at the past $1/(1 - \beta)$ times. In our experiment, we obtained a relatively stable result when β is set to 0.9999. This step is performed after each backpropagation, and all parameters with a gradient are involved.

5.2. Data augmentation

Datasets in industrial manufacturing contain relatively few images, which means that they require more effective data augmentation measures. In addition, the model performance is closely related to the batch size during training. A larger batch size means that the BN layer calculates more images at a time, thus avoiding the interference of random disturbances. However, a large batch size also means high GPU occupation, which is counter to the standards of industrial application.

To solve these problems, we adopt the mosaic data augmentation method. Mosaic data augmentation cuts and splices four randomly selected images stochastically into one image, retaining the visible targets and deleting the invisible targets. Four examples of mosaic data augmentation are exhibited in Fig. 9. By combining four pictures into one, we pass four images into the model at one time, which is equivalent to increasing the batch size. In this way, we avoid using a large batch size and reduce the hardware requirements in a disguised way.

In addition, mosaic data augmentation greatly enriches the background of the detected target and enhances the robustness of the model. Table 1 shows the effect of mosaic data enhancement.

In addition, we aim to resize the input images to a fixed value of a multiple of 32 between 320 and 640 (since the backbone halves the size of the images five times, the sizes of the input images must be multiples of 32). Experiments (see Table 2) demonstrate that the model becomes more effective as the image size increases. However, it will put a great deal of pressure on the hardware if the input image size is over 640 (the total GPU memory of GTX 1080Ti is 11 Gb). We adopt a multi-scale strategy to randomly select a multiple of 32 between 320 and 640 as the size of the input image. Compared with one-scale training, multi-scale training reduces the hardware requirements, and it not only strengthens the robustness of the model to images of different sizes but also improves the mAP index of the model.

For traditional methods, we use augmentations in the HSV domain to randomly change the hue, saturation, and value. In addition, we append normalization, random flips, and Gaussian noise to the augmentation method list.

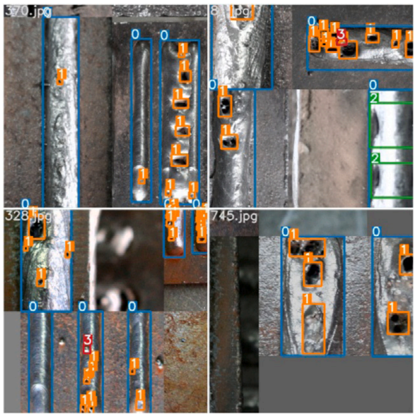


Fig. 9. Examples of mosaic data augmentation.

Table 1
Experiments with mosaic data augmentation.

Training	mAP@0.5
With mosaic	**94.5**
Without mosaic	90.1

Table 2
MAP for different input sizes.

Type	Size	mAP@0.5	GPU memory
One-scale	320	82.2	3.63 Gb
	384	85.3	4.28 Gb
	448	87.3	5.19 Gb
	512	87.8	6.56 Gb
	576	89.5	7.77 Gb
	640	90.9	10.71 Gb
Multi-scale	320–640	**94.5**	9.62 Gb

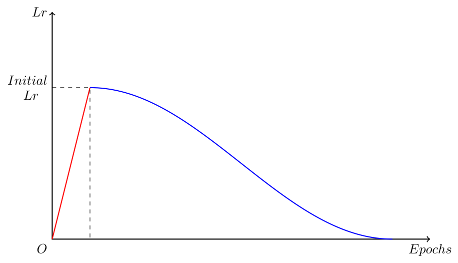


Fig. 10. Visualization of cosine decay with a warm-up.

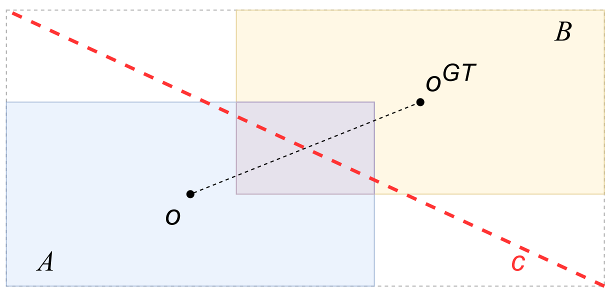


Fig. 11. CIoU.

5.3. Learning rate schedule

The learning rate (Lr) is one of the most critical parameters because it directly determines the speed and accuracy of training. A large Lr speeds up the training process and helps the model cross the local minimum, but it is difficult for it to reach the optimal global value. In contrast, a small Lr slows down the training and makes it very easy for the model to be trapped in a local minimum, but it is capable of ultimately reaching the optimal global value.

Step decay is the most commonly employed Lr schedule in general training; it uses a larger Lr for the first decades of epochs and reduces the Lr whenever the loss does not decrease. However, it is difficult to determine whether the final Lr is small enough to reach the global optimum. For our training strategies, we adopt cosine decay [58] with a warm-up.

Fig. 10 shows that, according to the warm-up, the Lr increases linearly from 0 to the established initial Lr value and then begins to decline according to the rule of the cosine function: first, it drops slowly, then approximately linearly, and finally, it drops slowly close to 0. Assuming that the total batch number to be cycled is T and that at batch t, the current Lr can be expressed as Eq. (4) (ignoring the warm-up phase):

$$\eta_t = \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)\eta \tag{4}$$

where η is the initial learning rate.

By following cosine decay, the Lr maintains a slow decline rate compared to step decay, which could speed up the training process in the initial phase and fine-tune the value of the weights at the end. Moreover, the model approaches the optimal global solution simply due to having a sufficiently small Lr at the later stage of training. The warm-up phase is designed to avoid initial oscillations.

5.4. Loss function

The value of the loss function represents the gap between the current model and the ideal state and indicates the direction for training. An applicable loss function accelerates the convergence process of the training. We define three loss functions based on the objective of locating and distinguishing different kinds of defects: CIoU loss [59], objectness loss, and classification loss.

The CIoU loss function indicates how far the prediction bounding boxes are from the ground truth (GT). The classical IoU loss function represents only the intersection area of the prediction bounding boxes and the GT. Thus, we add the distance of the centre point and the ratio of the side lengths into the loss function by using the CIoU.

As shown in Fig. 11, assuming that one prediction bounding box is A, the corresponding GT is B, and c is the diagonal length of the minimum enclosing rectangle of A and B, the formulas for $Loss_{CIoU}$ can be expressed as:

$$Loss_{CIoU} = 1 - IoU + R_{CIoU} \tag{5}$$

$$IoU = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{6}$$

$$R_{CIoU} = \frac{\rho^2(b, b^{GT})}{c^2} + \alpha v \tag{7}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{8}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{GT}}{h^{GT}} - arctan\frac{w}{h}\right)^2 \tag{9}$$

where $\rho(\cdot)$ is the Euclidean distance between two points and R_{CIoU} is the compensation factor for the position and shape.

In addition, the confidence loss $Loss_{obj}$ indicates whether there is an object. The classification loss $Loss_{cls}$ indicates whether the type of target detected is correct. The total loss function is expressed as Eqs. (10) (11) (12):

$$Loss = \lambda_{CIoU} Loss_{CIoU} + \lambda_{obj} Loss_{obj} + \lambda_{cls} Loss_{cls} \tag{10}$$

$$\begin{aligned} Loss_{obj} = &-\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}\left[\hat{C}_i^j \log(C_i^j) + (1 - \hat{C}_i^j)\log(1 - C_i^j)\right] \\ &-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}\left[\hat{C}_i^j \log(C_i^j) + (1 - \hat{C}_i^j)\log(1 - C_i^j)\right] \end{aligned} \tag{11}$$

$$Loss_{cls} = -\sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c\in classes}\left(\left[\hat{P}_i^j \log(P_i^j) + (1 - \hat{P}_i^j)\log(1 - P_i^j)\right]\right) \tag{12}$$

When each epoch ends, we test our model on the validation dataset to calculate the mAP index and the values of the various loss functions. By observing the indicators, we determine the training situation and whether there is overfitting or underfitting.

5.5. Anchors

As mentioned above, the prediction bounding boxes strictly correspond to anchors. Anchors of the appropriate size speed up the training and improve the fitness of the objectives. To make the size of the anchors consistent with our custom dataset, we abandon YOLOv4's original anchors and generate new ones through the K-means method.

5.6. Transfer learning

Due to the high cost of obtaining defect sample images in WAAM, we only have 680 training set images and 80 verification set images. An insufficient dataset easily results in overfitting or underfitting during the training of a deep learning network. To solve this problem, we adopt the approach of transfer learning.

The COCO [15] dataset, which has over 300,000 tagged images with over 1.5 million targets, is sufficient for training an object detection model. We first train our model based on the COCO dataset to obtain pretrained weights and then continue the training process with our custom dataset. The comparison between training from scratch and transfer learning is shown in Fig. 12.

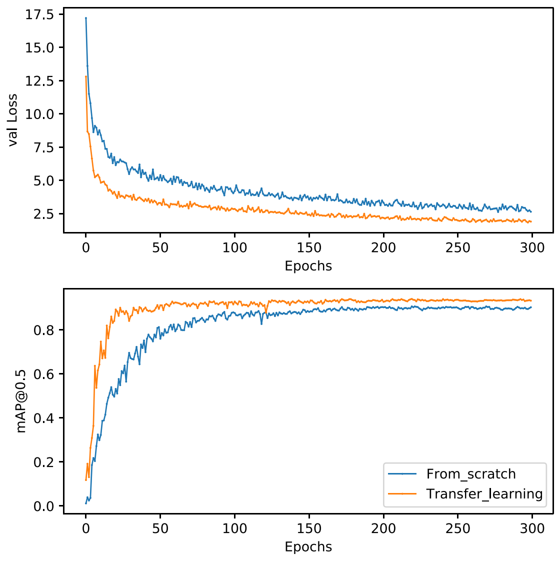


Fig. 12. Transfer learning.

Table 3
Dataset details.

Type	Num
Images	760
Weld	1975
Surface pore	4420
Groove	327
Slag inclusion	1861

It is evident that transfer learning accelerates the training process significantly and improves the performance of the final model. Specifically, the mAP reaches 94.5 with transfer learning for 126 epochs but only 90.7 when training from scratch for 236 epochs.

6. Experiments

This section primarily introduces and summarizes the experimental conclusions. First, we illustrate our computer environments and major hyperparameters. Second, we exhibit the training results of our model and its comparison with other models to verify the innovation of our approach.

6.1. Dataset

In this research, the dataset consists of 760 tagged images and 8583 tagged targets. Table 3 gives the details of our dataset. Among the images in the shuffled dataset, we select 10% as a test set to perform cross-validation and demonstrate the feasibility and superiority of our method.

6.2. Experimental configurations

The computer environment on which our experiments are based is shown in Table 4. In the training process, we used the Kaiming normal to initialize the weights and adopt an optimizer for stochastic gradient descent (SGD), with momentum and weight decay assigned as 0.937 and 0.000332, respectively. We set the total number of training epochs to 300 and the batch size to 6. The initial Lr η is set to 0.01, and the final Lr is set to 0.0005. The momentum for the BN layers is assigned as 0.03. For the loss function, we set λ_{CIoU}, λ_{obj}, and λ_{cls} to 2.52, 102, and 1.51, respectively. In addition, we fuse the BN layers and convolution layers in the prediction phase to accelerate the prediction process.

Table 4
Computer environment.

Item	Value
System	Ubuntu 18.04
CPU	i7 7700K
GPU	Nvidia GeForce GTX 1080Ti
RAM	32 Gb
CUDA	10.2
PyTorch	1.5.0
Torchvision	0.6

Table 5
Comparison between models.

Model	Type	mAP@0.5	FPS
Ours	One-Stage	**94.5**	42
YOLOv4	One-Stage	72.8	**45**
SSD300	One-Stage	52.9	42
RetinaNet	One-Stage	65.6	10
Faster-RCNN	Two-Stage	67.0	5
EfficientDet-D0	One-Stage	71.1	36
EfficientDet-D1	One-Stage	83.6	30
EfficientDet-D2	One-Stage	93.1	26

Table 6
A detailed comparison with YOLOv4 for mAP@0.5.

Category	Ours	YOLOv4	Improved by
Total	**94.5**	72.8	29.8%
Weld	**99.8**	99.7	0.1%
Surface pore	**91.7**	66.6	37.7%
Groove	**93.7**	76.0	23.3%
Slag inclusion	**92.6**	48.8	89.8%

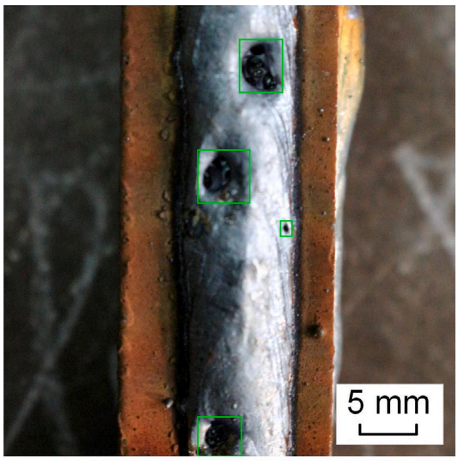
6.3. Experimental results

To verify the improvement of YOLO-attention, various comparison models were tested during the experiment. A Friedman test was also performed on the results, which yielded a probability p less than 0.01. Table 5 displays the experimental results of the proposed model on our test set, where all processes were performed under the same experimental conditions, such as transfer learning based on the COCO dataset. As shown in Table 5, compared to the original YOLOv4, our model improved the mAP by 29.8% with a slight reduction in the FPS. From the detailed information in Table 6, the effect of our model compared to YOLOv4 shows little difference in inspecting simple welding beads but is improved by 37.7%, 23.3%, 78.8% in the other three more difficult inspections, illustrating that the model can better inspect small and intensive targets after our optimization. For SSD300, our model improved by 78.6% while ensuring the same FPS. In addition, our model also has a 44.1% lead on RetinaNet [60].

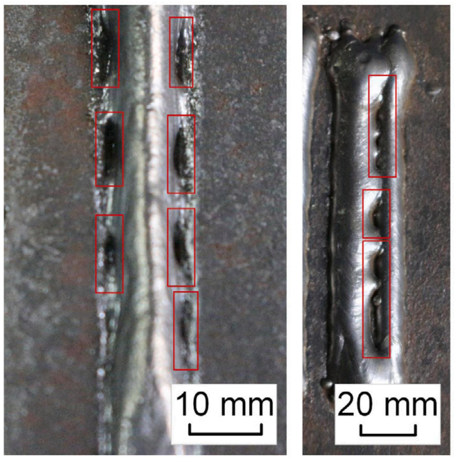
Regarding the two-stage detection model, its detection efficiency does not satisfy the needs of industrial applications due to its complexity. However, we still test the performance of Faster-RCNN on our dataset. The experimental results indicate that the efficiency of our model is eight times greater and the mAP improves by 41.0%.

We also experiment on the state-of-the-art model EfficientDet and list the results. Since the demand for real-time detection is at least 30 FPS, only EfficientDet-D0 and EfficientDet-D1 meet our requirements. Our model still has advantages in terms of accuracy and speed. Only for the lower efficiency requirement does the EfficientDet approach the accuracy of our model. In addition, our model requires less time and GPU memory during training; EfficientDet-D1 takes 30 h for 300 epochs, occupying 10.51 Gb GPU memory, while ours takes only 7 h and 9.62 Gb GPU memory.

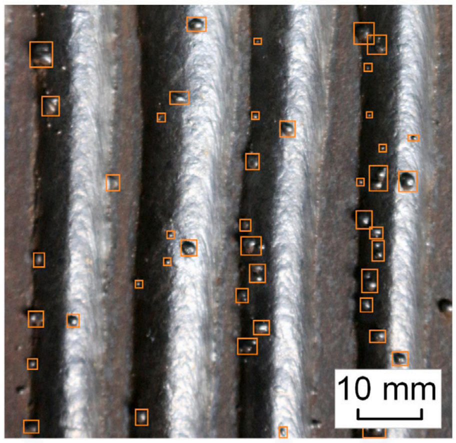
For practical industrial applications, we select several complex images with small and intensive defects for testing and display the prediction results in Fig. 13. Our model still has an excellent detection effect for intensive small targets and correctly excludes confusing samples that are not on the weld bead.



(a) surface pores



(b) grooves



(c) slag inclusions

Fig. 13. Defect detection results in WAAM.

6.4. *Ablation experiments*

Our experimental results demonstrate the effectiveness of the channel-wise attention mechanism in detection, as shown in Table 7.

To validate our modification of activation functions, we tested different activation functions and compared their performance in Table 8. In particular, we found that the Mish [61] and Swish [62] activation functions significantly increase the occupation of GPU memory, which increases the training difficulty and is not suitable for our industrial application.

The SPP operation is not a necessary step in extracting features and directly increases the GPU computation time. However, it effectively improves the mAP of the model because it extracts more sophisticated features with different receptive fields to improve the sensitivity to small targets. We carried out ablation experiments, and the results are shown in Table 9.

After applying the EMA to the training and then tracking the weight, as shown in Fig. 14, we conclude that the fluctuation range of the weight updating curve becomes smoother, indicating that the influence of the various interfering factors decreases. In the later stage of training, the EMA can also reduce such oscillations and make the model converge to the optimal global solution, which effectively improves the inference performance without increasing the computational burden of the model (Table 10).

Table 7
Performance of different backbones.

Backbone	mAP@0.5	FPS
DarkNet53-attention	**94.5**	42
CSPDarkNet53	87.9	45
DarkNet53	86.7	43

Table 8
Comparison between different activation functions.

Activation function	mAP@0.5
PReLU	**94.5**
Leaky ReLU	93.6
Mish	92.1
Swish	92.0

Table 9
Tests with SPP.

Model	mAP@0.5	FPS
with 3 SPPs	**94.5**	42
with 1 SPP	89.2	43
no SPP	84.0	**44**

Table 10
Experiments on the EMA.

Training	mAP@0.5
With EMA	**94.5**
Without EMA	86.1

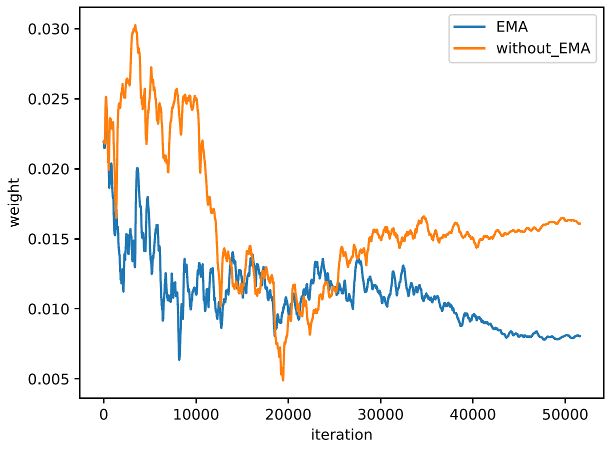
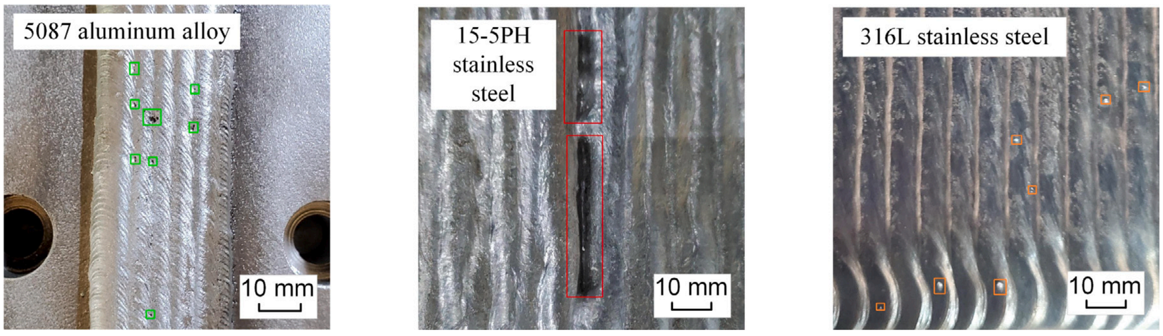


Fig. 14. Visualization of the EMA.



(a) surface pores (b) grooves (c) slag inclusions

Fig. 15. Defects in multi-pass of different materials.

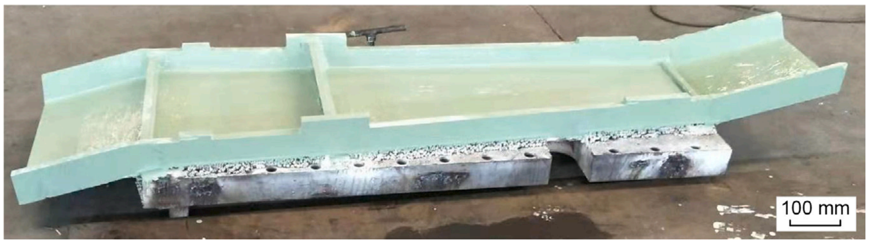


Fig. 16. Pylon beam of civil aircraft fabricated by additive manufacturing. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Finally, in order to rigorously demonstrate the effectiveness of the optimization method in this paper, we conducted 10 independent experiments on the model before and after optimization and analysed the experimental results using ANOVA. The p obtained was less than 0.01. The above statistical analysis proved that the optimization method in this paper has a very significant effect.

6.5. Validation

In further research, the YOLO-attention method is used to be trained based on the defect data of multi-pass deposition of different materials and applied in defect detection. Fig. 15 shows the detection results. Fig. 16 shows the pylon beam of a civil aircraft fabricated by additive manufacturing. The green coatings are antioxidation coatings for heat treatment. The material is 15–5PH stainless steel. To prevent defects, the detection system detects in real time in the whole additive manufacturing process. Through defect detection of multi-pass deposition and parts, it is proven that this method has wide application prospects in the field of additive manufacturing.

7. Conclusion

In this paper, we introduce and propose an improved model based on YOLOv4 to address the difficult problems of defect detection in WAAM. First, we describe the whole detection process of the defect detection system. Second, we present the backbone network, SPP, and FPN of our model and illustrate the strategies and tricks used in the training process. Finally, we demonstrate the effect and robustness of our detection system and compare it with other detection models to show its advantages.

The three techniques adopted in our model greatly improve the object detection capability for small and dense defects in WAAM. First, the channel-wise attention mechanism in the backbone network effectively improves the detection effect and enhances the model by 7.7 mAP while only reducing the speed by 3 FPS. Second, the multiple-SPP structure in the FPN extracts additional information with receptive fields of different sizes, which improves the model by 10.4 mAP. Third, the EMA strategy used in the training process improves the accuracy by 8.4 mAP without increasing the amount of calculation in the inference stage.

In summary, our model obtained an mAP of 94.5 on the test set, and the calculation speed reached 42 FPS with an NVIDIA GeForce GTX 1080Ti, which is sufficient for a real-time defect detection system. In conclusion, the experiments indicate that our proposed model has a high enough precision and efficiency to be applied in practical WAAM industry production.

CRediT authorship contribution statement

Wenhao Li: Methodology, Investigation, Writing – original draft. **Haiou Zhang:** Funding acquisition. **Guilan Wang:** Project administration. **Gang Xiong:** Data curation. **Meihua Zhao:** Validation. **Guokuan Li:** Methodology. **Runsheng Li:** Formal analysis, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants U1909218, 61872365; Research and Development of Laser Repair Technology and Equipment, China for Landing Gear and Other Key Metal Parts of Transport Aircraft, Hubei Province Technology Innovation Special Key Project (Grand no. 2019AAA003); Scientific Instrument Developing Project of the Chinese Academy of Sciences (CAS) under Grant YZQT014; Guangdong Basic and Applied Basic Research Foundation under Grant 2021B1515140034.

References

[1] B. Wu, Z. Pan, D. Ding, D. Cuiuri, H. Li, J. Xu, J. Norrish, A review of the wire arc additive manufacturing of metals: Properties, defects and quality improvement, J. Manuf. Process. 35 (2018) 127–139.
[2] C.R. Cunningham, J.M. Flynn, A. Shokrani, V. Dhokia, S.T. Newman, Invited review article: Strategies and processes for high quality wire arc additive manufacturing, Addit. Manuf. 22 (2018) 672–686.
[3] W.E. Frazier, Metal additive manufacturing: A review, J. Mater. Eng. Perform. 23 (2014) 1917–1928.
[4] I. Gibson, D.W. Rosen, B. Stucker, et al., Additive Manufacturing Technologies, Vol. 17, Springer, 2014.
[5] R. Li, H. Zhang, F. Dai, C. Huang, G. Wang, End lateral extension path strategy for intersection in wire and arc additive manufactured 2319 aluminum alloy, Rapid Prototyp. J. (2019).
[6] J. Mazumder, D. Dutta, N. Kikuchi, A. Ghosh, Closed loop direct metal deposition: Art to part, Opt. Lasers Eng. 34 (2000) 397–414.
[7] P. Wanjara, M. Brochu, M. Jahazi, Electron beam freeforming of stainless steel using solid wire feed, Mater. Des. 28 (2007) 2278–2286.
[8] A. Sreenathbabu, K.P. Karunakaran, C. Amarnath, Statistical process design for hybrid adaptive layer manufacturing, Rapid Prototyp. J. (2005).
[9] S.C. Joshi, A.A. Sheikh, 3D printing in aerospace and its long-term sustainability, Virtual Phys. Prototyp. 10 (2015) 175–185.
[10] R. Li, G. Wang, Y. Ding, S. Tang, X. Chen, F. Dai, R. Wang, H. Song, H. Zhang, Optimization of the geometry for the end lateral extension path strategy to fabricate intersections using laser and cold metal transfer hybrid additive manufacturing, Addit. Manuf. 36 (2020) 101546.
[11] R. Li, G. Wang, X. Zhao, F. Dai, C. Huang, M. Zhang, X. Chen, H. Song, H. Zhang, Effect of path strategy on residual stress and distortion in laser and cold metal transfer hybrid additive manufacturing, Addit. Manuf. 46 (2021) 102203.
[12] A. Taşdemir, S. Nohut, An overview of wire arc additive manufacturing (WAAM) in shipbuilding industry, Ships Offshore Struct. (2020) 1–18.
[13] R.H. Lee, The other electrical hazard: Electric arc blast burns, IEEE Trans. Ind. Appl. IA-18 (1982) 246–251.
[14] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: Optimal speed and accuracy of object detection, 2020, arXiv preprint arXiv:2004.10934.
[15] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, C.L. Zitnick, Microsoft coco captions: Data collection and evaluation server, 2015, arXiv preprint arXiv:1504.00325.
[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: Single shot multibox detector, in: European Conference on Computer Vision, Springer, 2016, pp. 21–37.
[17] J. Li, Z. Su, J. Geng, Y. Yin, Real-time detection of steel strip surface defects based on improved yolo detection network, IFAC-PapersOnLine 51 (2018) 76–81.
[18] A.R. Nassar, T.J. Spurgeon, E.W. Reutzel, Sensing defects during directed-energy additive manufacturing of metal parts using optical emissions spectroscopy, in: Solid Freeform Fabrication Symposium Proceedings, University of Texas Austin, TX, 2014, pp. 278–287.
[19] S. Everton, P. Dickens, C. Tuck, B. Dutton, Evaluation of laser ultrasonic testing for inspection of metal additive manufacturing, in: Laser 3D Manufacturing II, Vol. 9353, International Society for Optics and Photonics, 2015, 935316.
[20] R. Ren, T. Hung, K.C. Tan, A generic deep-learning-based approach for automated surface inspection, IEEE Trans. Cybern. 48 (2017) 929–940.
[21] Y.-G. Cen, R.-Z. Zhao, L.-H. Cen, L.-H. Cui, Z.-J. Miao, Z. Wei, Defect inspection for TFT-LCD images based on the low-rank matrix reconstruction, Neurocomputing 149 (2015) 1206–1215.
[22] S. Susan, M. Sharma, Automatic texture defect detection using Gaussian mixture entropy modeling, Neurocomputing 239 (2017) 232–237.
[23] X. Bai, Y. Fang, W. Lin, L. Wang, B.-F. Ju, Saliency-based defect detection in industrial images by using phase spectrum, IEEE Trans. Ind. Inf. 10 (2014) 2135–2145.
[24] R. Borwankar, R. Ludwig, An optical surface inspection and automatic classification technique using the rotated wavelet transform, IEEE Trans. Instrum. Meas. 67 (2018) 690–697.
[25] G.-H. Hu, Automated defect detection in textured surfaces using optimal elliptical Gabor filters, Optik 126 (2015) 1331–1340.
[26] J.A. Tsanakas, D. Chrysostomou, P.N. Botsaris, A. Gasteratos, Fault diagnosis of photovoltaic modules through image processing and canny edge detection on field thermographic measurements, Int. J. Sustain. Energy 34 (2015) 351–372.
[27] C. Tastimur, H. Yetis, M. Karaköse, E. Akin, Rail defect detection and classification with real time image processing technique, Int. J. Comput. Sci. Softw. Eng. 5 (2016) 283.
[28] C. Jian, J. Gao, Y. Ao, Automatic surface defect detection for mobile phone screen glass based on machine vision, Appl. Soft Comput. 52 (2017) 348–358.
[29] K.-L. Mak, P. Peng, K.F.C. Yiu, Fabric defect detection using morphological filters, Image Vis. Comput. 27 (2009) 1585–1592.
[30] T. Kalaiselvi, P. Nagaraja, A rapid automatic brain tumor detection method for MRI images using modified minimum error thresholding technique, Int. J. Imaging Syst. Technol. 1 (2015) 77–85.
[31] M. Win, A.R. Bushroa, M.A. Hassan, N.M. Hilman, A. Ide-Ektessabi, A contrast adjustment thresholding method for surface defect detection based on mesoscopy, IEEE Trans. Ind. Inf. 11 (2015) 642–649.
[32] X.-c. Yuan, L.-s. Wu, Q. Peng, An improved Otsu method using the weighted object variance for defect detection, Appl. Surf. Sci. 349 (2015) 472–484.
[33] X. Li, B. Gao, W.L. Woo, G.Y. Tian, X. Qiu, L. Gu, Quantitative surface crack evaluation based on eddy current pulsed thermography, IEEE Sens. J. 17 (2016) 412–421.
[34] L. Wang, Y. Zhao, Y. Zhou, J. Hao, Calculation of flexible printed circuit boards (FPC) global and local defect detection based on computer vision, Circuit World (2016).
[35] N. Knezović, B. Dolšak, In-process non-destructive ultrasonic testing application during wire plus arc additive manufacturing, Adv. Produc. Eng. Manag. 13 (2018) 158–168.
[36] A. Lopez, R. Bacelar, I. Pires, T.G. Santos, J.P. Sousa, L. Quintino, Non-destructive testing application of radiography and ultrasound for wire and arc additive manufacturing, Addit. Manuf. 21 (2018) 298–306.
[37] A. Chabot, N. Laroche, E. Carcreff, M. Rauch, J.-Y.. Hascoët, Towards defect monitoring for metallic additive manufacturing components using phased array ultrasonic testing, J. Intell. Manuf. (2019) 1–11.
[38] V.P. Vavilov, S.S. Pawar, A novel approach for one-sided thermal nondestructive testing of composites by using infrared thermography, Polym. Test. 44 (2015) 224–233.
[39] M. Mahmoudi, A.A. Ezzat, A. Elwany, Layerwise anomaly detection in laser powder-bed fusion metal additive manufacturing, J. Manuf. Sci. Eng. 141 (2019).
[40] C. Zhang, M. Gao, C. Chen, X. Zeng, Spectral diagnosis of wire arc additive manufacturing of Al alloys, Addit. Manuf. 30 (2019) 100869.
[41] Z. Zhang, W. Ren, Z. Yang, G. Wen, Real-time seam defect identification for Al alloys in robotic arc welding using optical spectroscopy and integrating learning, Measurement 156 (2020) 107546.
[42] M. Tan, Q.V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, 2019, arXiv preprint arXiv:1905.11946.
[43] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.
[44] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, et al., DeepID-Net: Object detection with deformable part based convolutional neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2016) 1320–1334.
[45] A. Diba, V. Sharma, A. Pazandeh, H. Pirsiavash, L. Van Gool, Weakly supervised cascaded convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 914–922.
[46] X. Tao, D. Zhang, W. Ma, X. Liu, D. Xu, Automatic metallic surface defect detection and recognition with convolutional neural networks, Appl. Sci. 8 (2018) 1575.
[47] W.-Y. Lin, C.-Y. Lin, G.-S. Chen, C.-Y. Hsu, Steel surface defects detection based on deep learning, in: International Conference on Applied Human Factors and Ergonomics, Springer, 2018, pp. 141–149.
[48] S. Ren, K. He, R. Girshick, J. Sun, Faster r-CNN: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[49] V. Pandiyan, P. Murugan, T. Tjahjowidodo, W. Caesarendra, O.M. Manyar, D.J.H. Then, In-process virtual verification of weld seam removal in robotic abrasive belt grinding process using deep learning, Robot. Comput.-Integr. Manuf. 57 (2019) 477–487.
[50] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, R. Wang, DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection, Inform. Sci. (2020).
[51] J. Redmon, A. Farhadi, YOLO9000: Better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.
[52] J. Redmon, A. Farhadi, YOLOv3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.
[53] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[54] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125.
[55] M. Tan, R. Pang, Q.V. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790.
[56] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916.
[57] N. Bodla, B. Singh, R. Chellappa, L.S. Davis, Soft-NMS–improving object detection with one line of code, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5561–5569.
[58] I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with restarts. CoRR, 2016, arXiv preprint arXiv:1608.03983.

[59] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, D. Ren, Distance-IoU loss: Faster and better learning for bounding box regression, 2019, arXiv preprint arXiv:1911.08287.

[60] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.

[61] D. Misra, Mish: A self regularized non-monotonic neural activation function, 2019, arXiv preprint arXiv:1908.08681.

[62] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, 2017, arXiv preprint arXiv:1710.05941.