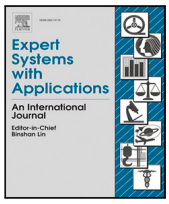
Expert Systems With Applications 267 (2025) 126212

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Gas pipeline defect detection based on improved deep learning approach

Ting Zhang [a], Cong Ma [a], Zhaoying Liu [a], Sadaqat ur Rehman [b,*], Yujian Li [c], Mohamad Saraee [b]

[a] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[b] School of Sciences, Engineering and Environment, University of Salford, Manchester, M5 4WT, UK
[c] School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China

ARTICLE INFO

Keywords:
Defect detection
Window self-attention
CrossFormer
Double attention decouple head
Multi-scale features

ABSTRACT

The working conditions of gas pipelines directly impact urban populations and factory operations. However, accurate and rapid detection of gas pipeline defects is challenging. To improve the accuracy of gas pipeline defect detection, we propose an improved RefineDet (Im-RefineDet) for gas pipeline defect detection, in which the improvement is carried out from the backbone network and the detection head. Specifically, to extract richer features, we design an improved CrossFormer as the backbone network. It first adopts a small convolutional cross-scale embedding layer to perform convolution, and then uses stripe window self-attention in vertical and horizontal directions in sequence to extract different features. In the detection head, we present a Double Attention Decouple Head (DADH) for classification and localization, enabling the model to perform independent optimization of the two branches. DADH employs spatial-aware and scale-aware attention to acquire multi-scale features, subsequently conducting classification and localization separately to derive final detection outcomes. Additionally, we apply channel pruning to the model to achieve a lightweight design, improving computational efficiency without significantly compromising detection performance. Experimental results, derived from an in-house developed gas pipeline defect image dataset, as well as two publicly available datasets — the NEU-DET dataset and the PCB dataset — demonstrate the effectiveness of the proposed Im-RefineDet. These results highlight its superior performance compared to state-of-the-art methods, further validating its robustness and adaptability across diverse scenarios. Specifically, the model achieves the mean Average Precision (mAP) of 92.6% on the gas pipeline defect image dataset, 77.8% on the NEU-DET dataset, and 99.2% on the PCB defect detection dataset.

1. Introduction

Gas pipelines are mainly used for gas transmission in daily life and factory operations (Arya, Jain, Yadav, Bisht, & Gautam, 2022). After reaching a certain service age, defects in shape and size can occur inside the gas pipeline due to chemical corrosion and physical damage. For defects, untimely detection and treatment can lead to safety accidents, which pose a threat to the healthy development of cities. Therefore, it is important to accurately understand the pipeline operating conditions (Mezher & Marble, 2024; Tian, Jiao, Liu, Ren, & Song, 2022). However, in practice, problems such as low defect image quality and diverse defect types cause manual interpretation to take relatively much time and energy (Layouni, Hamdi, & Tahar, 2017).

Recently, there have been some studies on defect detection methods based on deep learning. According to the processing flows, they belongs to two different types: two-stage models and single-stage models (Qi, Yang, & Zhong, 2020). Models of the former type first produce candidate boxes, and then do classification and localization on each candidate region to obtain the final detection result. For example, to detect the unobvious defects on printed circuit boards, Luo et al. presented a decoupled two-stage detection model by decoupling classification and localization tasks (Luo, Yang, Li, & Wu, 2021). Chen et al. presented a fabric defect detection model by combining Gabor filter with Faster RCNN (Chen et al., 2022). To detect small defects, Zeng et al. raised a balanced-feature pyramid network with atrous spatial pyramid pooling operation (Zeng et al., 2022).

For the later type, models realize the target detection with a regression procedure. They use only one network to analyze both the position and class information of the target simultaneously, directly outputing the detection results (Qi et al., 2020). For instance, Li et al. put forward to a defect detection method based on improved YOLOv4 (Li et al., 2023). It achieves higher accuracy on electric-arc additive manufacturing defect detection, as well as not decreasing the running speed.

* Corresponding author.
E-mail address: s.rehman15@salford.ac.uk (S. ur Rehman).

https://doi.org/10.1016/j.eswa.2024.126212
Received 22 August 2024; Received in revised form 24 November 2024; Accepted 16 December 2024
Available online 20 December 2024

Shafi et al. proposed a model based on improved SSD to detect the defects on the surface of hollow cylinders (Shafi, Mazahir, Fatima, & Ashraf, 2022). By integrating different channel attention and adaptive spatial features, Cheng et al. proposed a RetinaNet model, achieving good results in steel surface defect detection (Cheng & Yu, 2020). Zheng et al. first released an insulators defects detection dataset, and then provided a benchmark Network-FINet (Zhang et al., 2022).

Two-stage models perform dual regressions on anchor boxes, yielding higher accuracy. However, it requires saving the results of the first stage, needing more computing resources and time. Conversely, the single-stage models conduct only one regression on the anchor boxes with fewer computing resources and time. However, this comes at the cost of lower accuracy. Later on, Zhang et al. presented the RefineDet model by combining the two types of methods (Zhang, Wen, Bian, Lei, & Li, 2018). It maintains the same running efficiency as the single-stage methods, while the accuracy is better than the two-stage ones. However, when it is applied to defect detection, the diversity and complexity of defect features lead to low detection accuracy. There are mainly two reasons:

(1) Firstly, the features obtained from the backbone network are not rich enough. The RefineDet model uses VGG (Simonyan & Zisserman, 2014) or ResNet (He, Zhang, Ren, & Sun, 2016) to extract single-scale features, resulting in insufficient feature expression.

(2) Secondly, the two branches of the detection head share the same features, which is not conducive to detection. The focus of two tasks are different. The former relies more on the high-level abstract features, while the later are more dependent on the low-level features.

In response to the first problem, vision Transformer is widely used in computer vision tasks (Carion et al., 2020; Dosovitskiy et al., 2021; Li, Yao, Pan & Mei, 2022; Xie et al., 2021; Zheng et al., 2021; Zhu, Su, Lu, Li, Wang, & Dai, 2021) with its good feature extraction capabilities. With two special operations, the cross-scale embedding layer (CEL) and long short distance attention, CrossFormer can interact and fuse multi-scale features, achieving better results than other models on multiple tasks (Wang et al., 2022). However, the CrossFormer network has the following problems when performing gas pipeline defect detection. On the one hand, when the cross-scale embedding layer in CrossFormer uses a large convolutional kernel for convolution, the detailed information of the feature may be lost, leading the network difficult to capture the features of small-size defects. On the other hand, the window self-attention in CrossFormer cannot align defect features well, making it easy to focus on some irrelevant features.

To address the second issue, existing methods usually decouple the classification branch and the localization branch. Usually, the two branches use different networks (Ge, Liu, Wang, Li, & Sun, 2021; Song, Liu, & Wang, 2020; Wu et al., 2020). Consequently, they have a large number of model parameters and lower detection speed. Furthermore, the mutual influence of different scales of features are not fully utilized.

Consequently, an improved RefineDet model is proposed for gas pipeline defect detection. For the backbone network, an improved CrossFormer is presented to extract features. Two enhancements are implemented: on the one hand, the cross-scale embedding layer is decomposed, gradually decreasing the size of the feature maps to avoid the loss of defect feature information. On the other hand, stripe window self-attention is utilized to better highlight defective areas. In the detection head, two changes are introduced. First, spatial-aware attention and scale-aware attention are applied to obtain multi-scale features, enabling the network to detect defects at different scales. Second, a double attention decoupled head is designed to separate the classification and localization branches.

The main contributions of our work are:

(1) We propose an improved RefineDet for gas pipeline defect detection, in which the improved CrossFormer is used to extract features and the double attention decouple head is designed to handle classification and localization, respectively.

(2) We present a small convolutional cross-scale embedding layer, called SCCEL, which uses small convolutional kernels to retain more spatial information.

(3) We introduce a stripe window self-attention, named SWSA. The backbone network employs Stripes Windows Self-Attention (SWSA) in both vertical and horizontal orientations. Vertical SWSA uses taller-than-wide attention windows to capture vertical features, while horizontal SWSA uses wider-than-tall windows to focus on horizontal features. This combination enhances the network's ability to effectively capture features from multiple directions.

(4) We put forward a double attention decouple head, denoted as DADH, to decouple classification and localization branches. It uses spatial attention and scale attention to enhance the model's perception of targets.

(5) We conduct abundant of experiments on two datasets to verify the effectiveness of the proposed method.

The remaining of the paper is organized as follows. Section 2 briefly analyzes the related work, Section 3 describes in detail the improved RefineDet, Section 4 provides the experimental results on the pipeline defect dataset and the public dataset, as well as their corresponding analysis, and the last section concludes the work of this paper.

2. Related works

Our work is mainly related to three aspects, including target detection, vision Transformer and window self-attention.

2.1. Target detection based on deep learning

Currently, existing target detection models belong to either two-stage or single-stage. For the two-stage method, the representative models are regions with R-CNN series (Cai & Vasconcelos, 2018; Girshick, 2015; Mittal, Sharma, Singh, & Dhull, 2022; Ren, He, Girshick, & Sun, 2015; Sun et al., 2023). Among them, the first proposed model is R-CNN (Mittal et al., 2022). It includes three step: obtaining the candidate regions, extracting features from candidate regions, classifying with support vector machine. Some models have been developed to address the problems of RCNN taking up a lot of memory and slow calculation. For example, Girshick et al. put forward the Fast R-CNN (Girshick, 2015). Unlike R-CNN, it does classification with a neural network. By introducing the region proposal network, Ren et al. presented the Faster RCNN (Ren et al., 2015). It further accelerates the running speed. Based on the cascading idea, Cai et al. proposed Cascade RCNN (Cai & Vasconcelos, 2018). It uses multiple cascade sub-networks to gradually screen candidate anchors, which improves the detection accuracy. Based on the sparsity of the target, Sun et al. proposed Sparse RCNN (Sun et al., 2023). It builds an efficient and accurate end-to-end target detection model through the design of sparse prior boxes.

For the single-stage models, the common used are Single Shot Multibox Detector (SSD) (Liu et al., 2016) and You Only Look Once (YOLO) series (Bochkovskiy, Wang, & Liao, 2020; Jocher et al., 2022; Redmon, Divvala, Girshick, & Farhadi, 2016; Redmon & Farhadi, 2017, 2018). By extracting multi-level features, SSD is able to detect multi-scale targets, with relatively high detection accuracy and computational efficiency. YOLO series just use a network to simultaneously provide the location and category of the target. Among them, YOLOv1 is the first version of the YOLO series (Redmon et al., 2016). It first divides the input image into different grids, and then uses a convolutional neural network to obtain bounding boxes and categories for each grid. YOLOv2 uses the Darknet-19 network as the backbone network, and introduces a multi-scale input training strategy and anchor boxes mechanism to improve small object detection capabilities (Redmon & Farhadi, 2017). YOLOv3 conducts a series of improvements based on YOLOv2, including using deeper Darknet-53, applying the feature pyramid module to obtain multi-scale features, and then to prediction (Redmon & Farhadi, 2018).

In addition to the above two types of models, there are also some hybrid models that combine them (Zhang et al., 2018). For example, RefineDet first extract features with the backbone network, then applies the anchor refinement module to obtain candidate regions, and finally uses the target detection module to classify and localize the obtained candidate regions. It can be viewed as a combination of RPN and SSD.

Alongside these conventional models, Transformer-based object detection models, such as the Detection Transformer (DETR) (Carion et al., 2020), represent a significant advancement in the field. Unlike traditional methods that rely on predefined anchors or proposal networks, DETR treats object detection as a direct set prediction problem, utilizing a transformer architecture to model global relationships between objects and the image. Variants like DAB-DETR (Liu et al., 2022), DN-DETR (Li, Zhang, et al., 2022), and Deformable DETR (Zhu et al., 2021) have been introduced to address specific limitations of the original DETR model. DAB-DETR introduces dynamic anchor boxes to improve the model's ability to adapt to object size and shape, enhancing performance on small and irregularly shaped targets. DN-DETR integrates denoising training to speed up convergence and improve robustness, while Deformable DETR introduces deformable attention mechanisms to reduce computational overhead and better capture fine-grained spatial details, particularly for small and occluded objects. These transformer-based models have demonstrated state-of-the-art detection accuracy, particularly in scenarios with complex object relationships and cluttered environments.

Although two-stage models generally offer higher detection accuracy, they tend to be slower. Single-stage models, while faster, often require improvements in accuracy. Hybrid models like RefineDet offer a balance, providing high accuracy without significantly sacrificing speed. Similarly, transformer-based models such as DETR and its variants have introduced a new paradigm for achieving end-to-end object detection without relying on handcrafted proposal mechanisms. These models excel at modeling global object relationships and achieve state-of-the-art accuracy in complex scenes. However, DETR models also have notable drawbacks. The original DETR suffers from slow convergence during training and requires large datasets to fully exploit its potential. Variants like DAB-DETR, DN-DETR, and Deformable DETR address some of these issues, but the convergence issue is not entirely resolved. Therefore, in constructing our model, we base it on RefineDet, while incorporating insights from both traditional and modern approaches like Transformer-based methods to achieve a balance between accuracy and speed.

2.2. Vision transformer

Transformer is previously applied to natural language processing (Devlin, Chang, Lee, & Toutanova, 2018). Later on, Dosovitskiy et al. proposed Vision Transformer (ViT), and applied it to image classification (Dosovitskiy et al., 2021). Although ViT has got good results, it needs relatively high requirements on data scale and running equipment.

Subsequently, researchers proposed a variety of improved models (Gao, Zhang, Yang, & Zhou, 2022; Liu et al., 2021; Wang et al., 2022). For example, Liu et al. presented Swin Transformer, and applied it on image classification (Liu et al., 2021). It uses window self-attention to process large-size images in stages, achieving better classification accuracy. Gao et al. then put Swin Transformer on surface defect detection model (Gao et al., 2022). It uses a new window shift method to enhance the feature interaction between windows. Wang et al. built CrossFormer (Wang et al., 2022). First, it designs a cross-scale attention module. Second, it combines cross-scale attention with Transformer to extract features. It achieves higher results compared to other models in multiple tasks.

However, the large convolutional kernel of the cross-scale embedding layer in CrossFormer will lost the target information. Additionally, the window self-attention in CrossFormer cannot align defect features well, making it easy to focus on some irrelevant features.

To overcome the shortcomings of CrossFormer, we use small convolutional kernels in the cross-scale embedding layer to retain more target information, and utilize windows in different directions for self-attention to enhance the ability of feature extraction.

2.3. Window self-attention

Self-attention obtains a representation of a single sequence by computing the relationship between different positions in that sequence (Vaswani et al., 2017). Traditional self-attention requires calculating attention weights between all position pairs. This may result in excessive computational and storage costs. Therefore, the window self-attention was proposed (Ramachandran et al., 2019). It avoids global calculation of the entire input sequence by introducing window constraints in self-attention. In this approach, only certain areas of interest need to be focused on. For example, Swin Transformer uses window self-attention for information interaction within and across windows (Liu et al., 2021). Additionally, Axial-deeplab uses axial self-attention to deal with the feature map as a local self-attention window (Wang et al., 2020). CSWin Transformer uses cross-shaped window self-attention, which can be viewed as parallel multiple rows and columns of axial self-attention (Dong et al., 2022). Pale Transformer deals with a pale-shaped area with the pale-shaped self-attention (Wu, Wu, Tan, & Guo, 2022). Pale-shaped self-attention can be viewed as parallel multiple rows and columns of axial self-attention with certain intervals.

Although the window self-attention makes the model focus on different targets, it cannot perceive different directions. Indeed, the other type of window self-attention mentioned above have perception capabilities in different directions. However, since the attention scope is global in the horizontal or vertical direction, the aspect ratios of the self-attention window are too large, resulting in an imbalance in the information extracted by the window from two different directions. Furthermore, due to the different sensitivities to different directions, feature learning will produce biases, limiting the feature extracting ability.

Therefore, we propose a new window self-attention, namely stripe window self-attention. Through horizontal or vertical window self-attention, the interaction range of image tokens in the corresponding direction is increased. Besides, the appropriate aspect ratio of the window can better highlight the object defect area of interest.

3. Method

In this section, we first introduce the overall framework of Im-RefineDet, then describes the two improved parts (backbone network and detection head) in detail, and finally gives its loss function.

3.1. Overall architecture

Similar to RefineDet, there are four parts in the improved RefineDet, namely backbone network, neck module, anchor refinement module and target detection module. Fig. 1 displays the overall architecture of the Im-RefineDet.

Backbone Network: It is an improved CrossFormer network (Im-CrossFormer). Especially, we design an improved cross-scale embedding layer as well as a stripe window self-attention to fuse cross-scale features. Therefore, it can provide effective features for the subsequent anchor refinement module and target detection module.

Neck Module: It uses both high-level and low-level features to integrate contextual information, thereby improving detection accuracy. For feature maps of different scales, it gradually integrate features. Then, the fused multi-scale feature map is input to the anchor refinement module and target detection module, respectively.

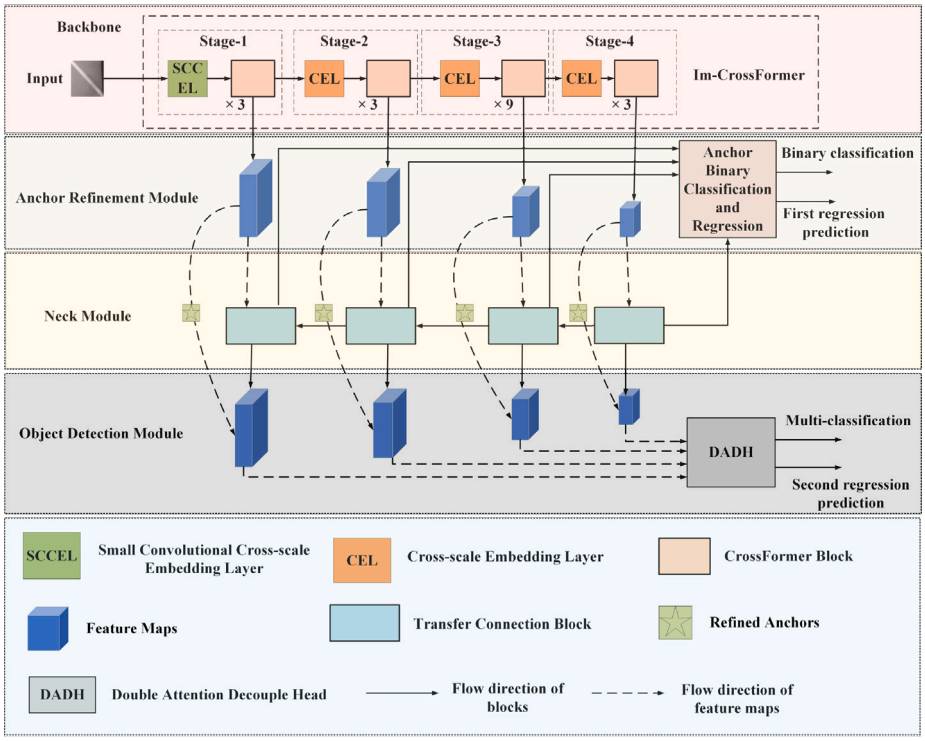


Fig. 1. The overall framework of the improved RefineDet. This enhanced model, fundamental to our methodology, prioritizes a meticulous approach towards identifying discrepancies and anomalies within gas pipeline structures. By focusing on refining the inherent defect detection mechanisms, our improved RefineDet (Im-RefineDet) meticulously scrutinizes image data, extracted from surveillance of pipeline networks, to effectively discern and localize potential defects. Consequently, this method facilitates not only the swift identification of defects, but also fosters a reliable mechanism to preemptively address potential pipeline failures, ensuring an augmented degree of safety and operational continuity within urban gas distribution networks and associated industrial entities.

Anchor Refinement Module: It is used to adjust the locations and sizes of the anchors to provide better initialization for the regression in the Object Detection Module (ODM). Specifically, n anchor boxes are associated with each regularly divided cell on the feature map. Initially, the position of each anchor box relative to its corresponding cell is fixed. At each feature map cell, the ARM predicts four offsets for the refined anchor boxes relative to the original tiled anchors. It also predicts two confidence scores that indicate the likelihood of the presence of foreground objects in those boxes, resulting in n refined anchor boxes at each feature map cell. To eliminate some redundant low-quality candidate anchor boxes, the ARM removes those with low confidence scores based on the predicted foreground confidence, thereby reducing the computational burden on the model.

Target Detection Module: It performs classification and localization on the candidate anchors to obtain the final detection result. First, it receives the candidate anchors filtered by the anchor refinement module and the fused multi-scale feature maps from the neck module. After convolutions, the feature maps are input into the double attention decouple head (DADH). Finally, the classification results and coordinate location results are predicted by DADH.

3.2. Im-CrossFormer

Since the cross-scale embedding layer realizes the convolution with large convolutional kernels, this may lost the detailed information of the target. Meanwhile, the window self-attention cannot align defect features well, making it easy to focus on some irrelevant features. Therefore, we propose an improved CrossFormer (Im-CrossFormer) as the backbone network. Fig. 2 manifests the structure of Im-CrossFormer.

As shown in Fig. 2, there are four stages in Im-CrossFormer. They are Stage-1, 2, 3, 4. There are two parts in each stage: a cross-scale embedding layer (or improved cross-scale embedding layer) as well as an Im-CrossFormer module. Among them, the cross-scale feature extraction layer of Stage-1 is small convolutional cross-scale embedding layer (SCCEL). For the remaining stages, we use the cross-scale embedding layer (CEL). For the four stages, the number of Im-CrossFormer blocks are 3, 3, 9 and 3, respectively.

Fig. 3 manifests the internal structure of three consecutive Im-CrossFormer blocks. From Fig. 3, we can find that each Im-CrossFormer block contains multilayer perceptron, window self-attention, layer normalization and cross-layer connection.

The first Im-CrossFormer block employs vertical stripe window self-attention, which focuses on capturing dependencies between features in the vertical direction at close distances. This helps the model understand local spatial relationships along the vertical axis.

The second Im-CrossFormer block uses horizontal stripe window self-attention, which focuses on capturing dependencies between features in the horizontal direction at close distances. This complements the vertical dependencies by modeling local relationships along the horizontal axis.

The third Im-CrossFormer block applies long-range self-attention (Wang et al., 2022), which captures dependencies between features that are spatially distant from each other. This helps the model understand global relationships across the entire feature map.

The cascading mechanism of these three blocks is designed to progressively refine the feature representation by integrating local and global dependencies. By stacking these specialized blocks, the model achieves a balanced learning of hierarchical spatial relationships, which enhances its capacity for accurate classification and localization tasks. This sequential design mirrors the multi-scale feature extraction mechanism seen in classical convolutional neural networks, but with a focus on leveraging the strengths of attention mechanisms in different spatial contexts.

They can be expressed as:

$$\boldsymbol{M}'_l = \text{V-SWSA}(\text{LayerNorm}(\boldsymbol{M}_{l-1})) + \boldsymbol{M}_{l-1} \tag{1}$$

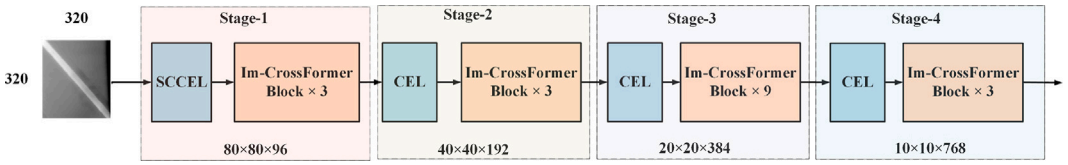


Fig. 2. The architecture of Im-CrossFormer.

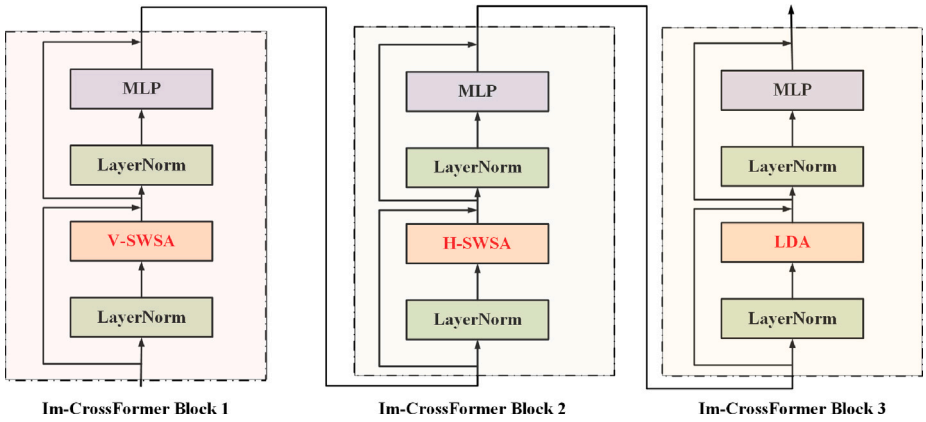


Fig. 3. Visualization of three successive Im-CrossFormer blocks.

$$\boldsymbol{M}_l = \mathrm{MLP}(\mathrm{LayerNorm}(\boldsymbol{M}'_l)) + \boldsymbol{M}'_l \tag{2}$$

$$\boldsymbol{M}'_{l+1} = \text{H-SWSA}(\mathrm{LayerNorm}(\boldsymbol{M}_l)) + \boldsymbol{M}_l \tag{3}$$

$$\boldsymbol{M}_{l+1} = \mathrm{MLP}(\mathrm{LayerNorm}(\boldsymbol{M}'_{l+1})) + M'_{l+1} \tag{4}$$

$$\boldsymbol{M}'_{l+2} = \mathrm{LDA}(\mathrm{LayerNorm}(\boldsymbol{M}_{l+1})) + \boldsymbol{M}_{l+1} \tag{5}$$

$$\boldsymbol{M}_{l+2} = \mathrm{MLP}(\mathrm{LayerNorm}(\boldsymbol{M}'_{l+2})) + \boldsymbol{M}'_{l+2} \tag{6}$$

where $\boldsymbol{M}_{l-1}$ means the input of the lth Im-CrossFormer block, $\boldsymbol{M}_l$, $\boldsymbol{M}_{l+1}$ and $\boldsymbol{M}_{l+2}$ denote the output of the lth, $l+1$-th and $l+2$-th Im-CrossFormer block, respectively; $MLP(\cdot)$ is the multilayer perceptron, and $LayerNorm(\cdot)$ stands for layer normalization. $V\text{-}SWSA(\cdot)$ and $H\text{-}SWSA(\cdot)$ indicate the vertical and horizontal stripe window self-attention, respectively. Their detailed introduction is given in Section 3 of 3.2. $LDA(\cdot)$ stands for long-distance self-attention.

3.2.1. SCCEL and CEL

To generate embeddings of the input of each stage, both the small convolutional cross-scale embedding layer (SCCEL) and the cross-scale embedding layer (CEL) use multi-scale convolution operations for down-sampling. For Stage-1, CEL in Crossformer uses four sets different sizes of convolution kernels, namely 4 × 4, 8 × 8, 16 × 16 and 32 × 32. All of the strides of the four groups of convolutional kernels are 4.

SCCEL decomposes the 8 × 8, 16 × 16 and 32 × 32 convolutional kernels in the above CEL into two 4 × 4, two 8 × 8, and two 16 × 16 convolution kernels, respectively.

In defect detection, images often suffer from issues like incomplete data and noise, making it harder to extract meaningful features. In the original CEL in Stage-1 of the CrossFormer network, large convolutional kernels with a stride of 4 are used to capture multi-scale features. However, this large stride results in significant downsampling of the feature map, which may cause the model to miss finer details of defects.

The SCCEL module improves on this by replacing the large kernels with two smaller kernels that have a stride of 2. This approach allows the model to retain more important defect information while still considering a larger area of the image. It is like using a smaller lens to zoom in and capture more detail, rather than using a large lens that might overlook smaller, important features.

Additionally, we introduce normalization layers (Layer Normalization, LN) and an activation function (Gaussian Error Linear Unit, GELU) to improve the training process and make the model more effective. These additions allow the network to better capture richer and more diverse multi-scale features.

By using smaller kernels, the SCCEL module increases the receptive field (the area the kernel "sees") more effectively, leading to a better balance between capturing fine details and a larger context from the image. This improvement ensures the model can handle defects better while maintaining efficiency in feature extraction.

Fig. 4 exhibits the process of SCCEL. For the input image, SCCEL first uses four sets of convolutional kernels for convolution to obtain four groups of feature maps. Then, it concatenates the feature maps according to the channel direction, obtaining the final embedding. Fig. 5 displays the detailed network structure diagram of SCCEL. In Fig. 5, GELU and LayerNorm represent Gaussian Error Linear Unit and Layer Normalization, respectively. Concat means concatenating the feature maps.

For the later three stages, CEL in Im-Crossformer uses two convolutional kernels of different sizes (2 × 2 and 4 × 4) to do convolution. The step size of the CEL of each stage is all set to 2, and the number of embeddings is a quarter of the original. Fig. 6 shows the CEL structure. The number of channels of the three stages are 192, 384 and 768, respectively.

3.2.2. Stripe window self-attention

The window-based attention mechanism reduces the computational cost of self-attention by limiting the interaction range of tokens. However, in pipeline scenarios, object features may have significant directionality, and the regular window self-attention, due to its square shape, may not align well with object features within the defined attention range, leading it to focus on background features that are unrelated to pipeline defect characteristics. Therefore, we propose the Stripes Windows Self-Attention (SWSA) module. SWSA enhances the interaction range of tokens in the corresponding direction through horizontal or vertical window self-attention, effectively highlighting the regions of interest with object defects without significantly increasing the computational burden. To further improve model performance, we enhance local feature extraction capabilities by concatenating SWSA in vertical and horizontal directions. SWSA is displayed in Fig. 7. For two adjacent Im-CrossFormer blocks, they have similar procedure: the

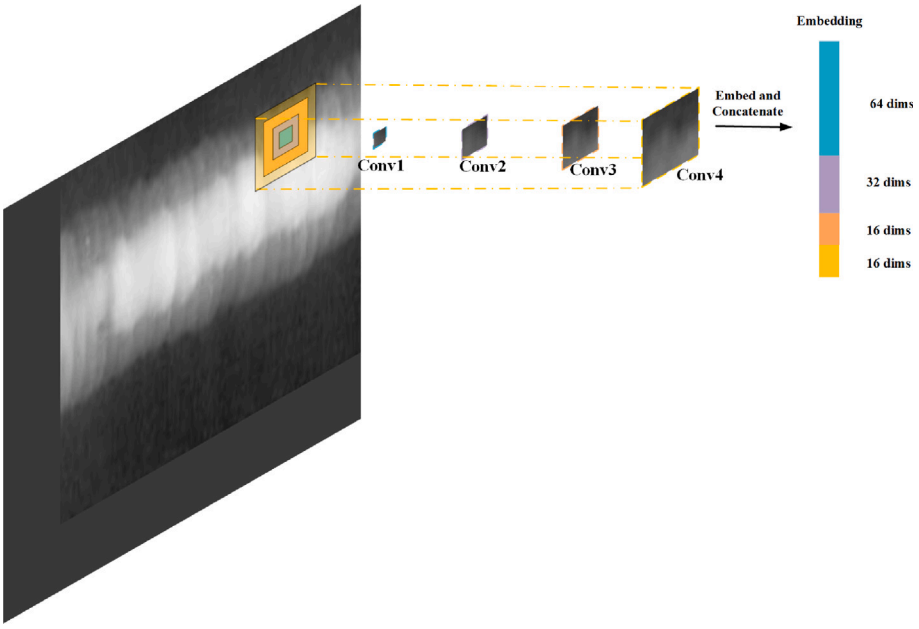


Fig. 4. Visualization of the SCCEL layer.

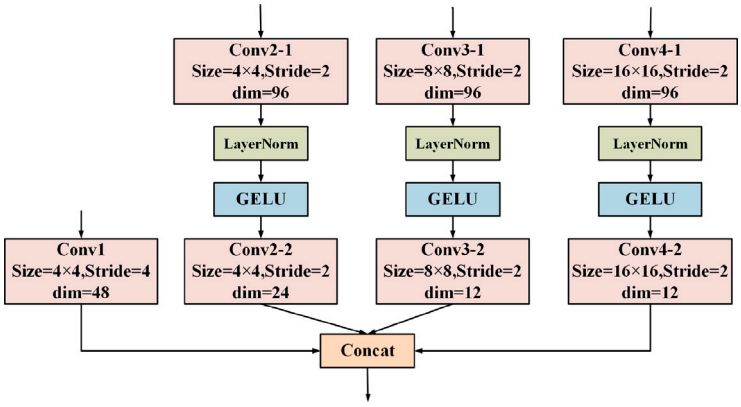


Fig. 5. The small convolutional cross-scale embedding layer.

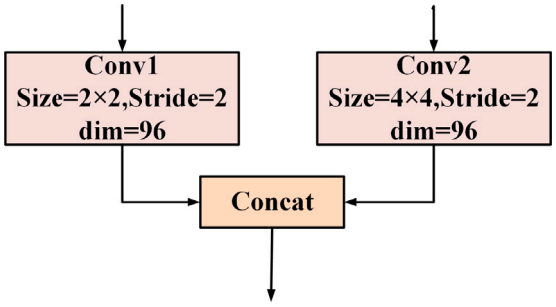


Fig. 6. Cross-scale embedding layer.

former divides the feature maps into several vertical stripe windows with height and width of 5 and 2, then performs self-attention calculation within the window; the later divides the feature maps into several horizontal stripe windows with height and width of 2 and 5, and then performs self-attention calculation within the window.

For feature map $\boldsymbol{M} \in R^{H\times W\times C}$, stripe window self-attention divides it into k non-overlapping $(G_h \times G_w) \times C$ windows, where G_h and G_w are the height and width of the window, respectively; k is equal to $\frac{H}{G_h} \times \frac{W}{G_w}$, and G_h and G_w are not equal. Stripe window self-attention can be expressed as:

$$\boldsymbol{M} = [\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_k] \tag{7}$$

$$\boldsymbol{Y}_i = \mathrm{Softmax}(\boldsymbol{M}_i \boldsymbol{W}^Q (\boldsymbol{M}_i \boldsymbol{W}^K)^T / \sqrt{C} + \boldsymbol{B}) \boldsymbol{M}_i \boldsymbol{W}^V \tag{8}$$

$$\mathrm{SWSA}(\boldsymbol{X}) = [\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_k] \tag{9}$$

where $\boldsymbol{M}_i \in R^{G_h\times G_w\times C}$ is the feature map, $k = \frac{H}{G_h} \times \frac{W}{G_w}$, $i = 1, 2, \ldots, k$. $\boldsymbol{W}^Q \in R^{C\times C}$, $\boldsymbol{W}^K \in R^{C\times C}$, and $\boldsymbol{W}^V \in R^{C\times C}$ respectively represent the projection matrix used to generate queries, keys and values during the self-attention calculation process; $\boldsymbol{Y}_i$ represents the feature map after self-attention calculation in the ith window; $\boldsymbol{B}$ is the relative position encoding matrix. When G_h is greater than G_w, the above stripe window self-attention is called vertical stripe window self-attention. When G_h is smaller than G_w, it is called horizontal stripe window self-attention.

3.2.3. Analysis of computational complexity of stripe window self-attention

In the Swin Transformer method (Liu et al., 2021), the computational complexity for a single attention window is given as:

$$\Omega(WSA) = 4HWC^2 + 2M^2HWC \tag{10}$$

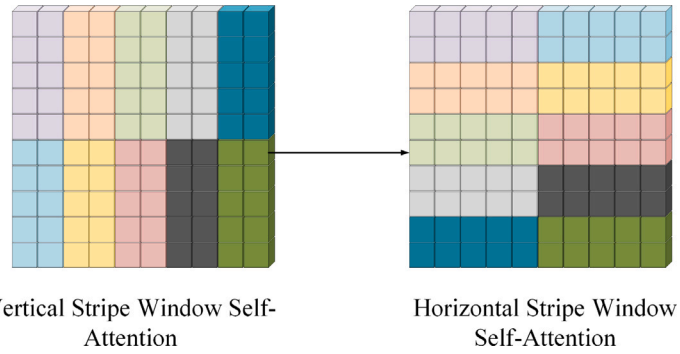


Fig. 7. The window self-attention in two successive Im-CrossFormer blocks.

where H and W represent the height and width of the feature map, C is the number of channels, and M denotes the size of each attention window.

In the SWSA module, a feature map with width W and height H is first divided into several equal-sized windows. Assuming each window has a width G_w and height G_h, the total number of windows is:

$$k = \frac{HW}{G_h G_w} \tag{11}$$

Multi-head self-attention is then applied within each window. Based on the attention window complexity formula from the Swin Transformer, the computational complexity for a single attention window in SWSA is:

$$\Omega(SWSA) = 4G_h G_w C^2 + 2(G_h G_w)^2 C \tag{12}$$

where G_h is the height and G_w is the width of the attention window, and C is the number of channels. Since there are k windows, the overall complexity for all attention windows in the feature map is:

$$\begin{aligned} k \times \Omega(SWSA) &= \frac{HW}{G_h G_w}(4G_h G_w C^2 + 2(G_h G_w)^2 C) \\ &= 2HWC(2C + G_h G_w) \end{aligned} \tag{13}$$

3.2.4. Lightweight channel pruning

As shown in Section 3.2.3, computational complexity is quadratically affected by both the number of channels and the window size. Since window sizes are significantly smaller than the number of channels, channel pruning is employed to reduce the overall computational load (Yamamoto & Maeno, 2018; Yu et al., 2022). By pruning channels, the number of channels is reduced from C to αC $(0 < \alpha < 1)$, and the complexity decreases by a factor of $(1 - \alpha^2)$. The complexity then becomes:

$$k \times \Omega(SWSA) = 2HW(2\alpha^2 C^2 + \alpha G_h G_w C) \tag{14}$$

where H and W represent the height and width of the feature map, G_h is the height and G_w is the width of the attention window, and C is the number of channels.

This represents a quadratic reduction. For example, if α=0.7 (pruning 30% of the channels), the computational load in the first part of the complexity formula decreases by approximately 51%.

The pruning method utilized is gradient-weighted pruning. This approach evaluates the importance of each channel based on the magnitude of the channel weights and gradient information, determining each channel's contribution to the loss function. By analyzing the sensitivity of the weights to the loss, channels that have less impact on the loss are pruned.

The pruning process begins with evaluating the importance of each channel. This is done by computing the gradient of the channel weights with respect to the loss function, which determines the contribution of each channel to the final prediction. The importance score for each channel is calculated as:

$$I = \left| W_i \times \frac{\partial L}{\partial W_i} \right| \tag{15}$$

where L represents the loss function, and $\frac{\partial L}{\partial W_i}$ is the gradient of the weights. This provides a measure of how critical each channel is to the overall model performance.

Once the importance scores are calculated, channels are ranked and the least impactful ones are pruned to reduce model complexity while preserving key features.

In multi-head self-attention, this pruning also reduces the dimensions of the query, key, and value matrices, as well as the number of attention heads, lowering computational cost without significantly affecting performance.

After pruning, fine-tuning is performed to recover or improve model accuracy by adapting to the reduced parameter set, ensuring efficiency without sacrificing.

3.3. Double attention decouple head

To make the model better detect defects with different scales, we present a double attention decouple head (DADH). On the one hand, we use spatial-aware and scale-aware attention to fuse multi-scale features so that the model has position and scale awareness. On the other hand, we decouple classification and localization branches to reduce the mutual influence between them. Fig. 8 displays the overall structure of the double attention decouple detection head and the specific details of its subparts. As shown in Fig. 8(a), we fuse the spatial-aware features with the scale-aware features. It is the remaining part of dynamic head after removing task-aware attention (Dai et al., 2021). Fig. 8(b) illustrates the structure of spatial- and scale-aware attention.

By generating offsets, the spatial-aware attention is designed to obtain the different spatial positions information. Given $\boldsymbol{M} \in R^{H \times W \times C}$, it can be defined as:

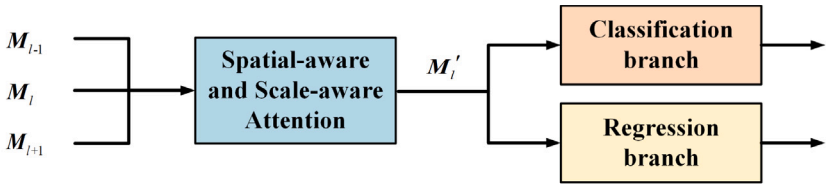
$$\pi_S(\boldsymbol{M}) \times \boldsymbol{M} = \sum_{k=1}^{K} w_k \boldsymbol{M}(p_k + \Delta p_k)\Delta m_k \tag{16}$$

where $\pi_S(\cdot)$ is a spatial attention function, w_k means the projection weight of the corresponding sampling point, Δp_k denotes the learned offset of the sampling point, Δm_k denotes the importance modulation scalar of Δp_k learned at the position, and $p_k + \Delta p_k$ stands for the new sampling points to be convolved.

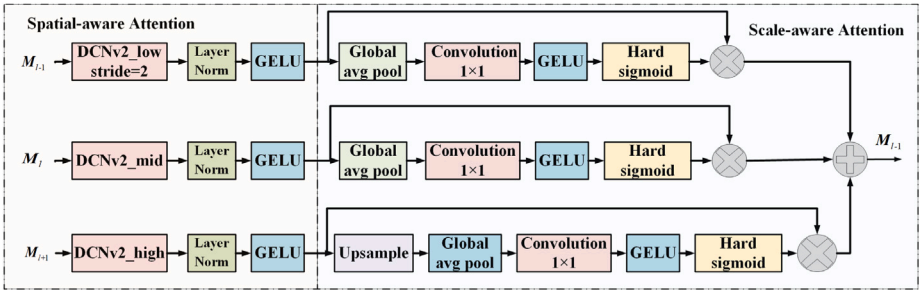
The role of scale-aware attention is to enhance the scale perception ability of the model by changing the expressive ability of targets of different scales. It is defined as:

$$\pi_L(\boldsymbol{M}) \times \boldsymbol{M} = \sigma(\mathrm{GELU}(f(\frac{1}{HWC}\sum_{HW,C} \boldsymbol{M}))) \times \boldsymbol{M} \tag{17}$$

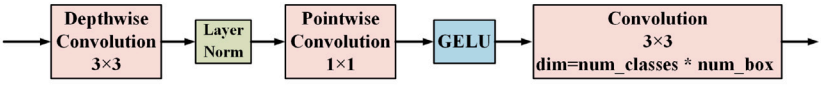
where $f(\cdot)$ is a linear function, $\pi_L(\cdot)$ represents a scale attention function, $\sigma(x) = \max(0, \min(1, \frac{x+1}{2}))$.



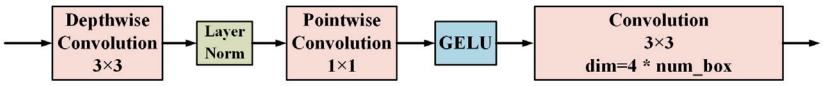
(a) The overall structure of DADH.



(b) The overall structure of double attention.



(c) Classification branch structure after decoupling.



(d) Localization branch structure after decoupling.

Fig. 8. The overall structure of DADH and its two branches.

Given three feature maps $\boldsymbol{M}_{i-1}$, $\boldsymbol{M}_i$ and $\boldsymbol{M}_{i+1}$, the two attentions (spatial-aware attention and scale-aware attention) can be computed as:

$$\boldsymbol{M}_i{}' = \pi_L((\pi_S(\boldsymbol{M}_{i-1}) \times \boldsymbol{M}_{i-1}) \times \boldsymbol{M}_{i-1}) + \pi_L((\pi_S(\boldsymbol{M}_i) \times \boldsymbol{M}_i) \times \boldsymbol{M}_i) + \pi_L((\pi_S(\boldsymbol{M}_{i+1}) \times \boldsymbol{M}_{i+1}) \times \boldsymbol{M}_{i+1}) \tag{18}$$

where $\pi_S(\cdot)$ and $\pi_L(\cdot)$ are the spatial attention function and the scale attention function, respectively.

Later on, the double attention decouple head uses a depth-wise separable convolution of 3 × 3 (Shaheed et al., 2022) to generate different feature maps, and then make classification and localization, respectively. Fig. 8(c) and Fig. 8(d) display the corresponding of the structures of two branches. Both branches include a 3 × 3 depth-wise convolution, a 1 × 1 point-wise convolution, layer normalization, and a GELU activation function. Finally, 3 × 3 convolutional kernels are used for classification and localization.

3.4. Loss function

The loss function of Im-RefineDet includes two parts: the loss of ARM and that of ODM. The localization loss of both modules is smoothing L1 loss (Ren et al., 2015), which is defined as:

$$L_r(p^u, v) = \sum_{k \in \{a,b,w,h\}} Smooth_{L_1}\left(p_k^u - v_k\right) \tag{19}$$

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2, |x| \le 1 \\ |x| - 0.5, |x| > 1 \end{cases} \tag{20}$$

where $v = (v_a, v_b, v_w, v_h)$, (v_a, v_b) is the center point coordinate axis of the true box, v_w and v_h are the width and height of the box; $p^u = (p_a^u, p_b^u, p_w^u, p_h^u)$, (p_a^u, p_b^u) is the center point coordinate axis of the predicted box, p_w^u and p_h^u are the width and height of the predicted box; $k \in \{a, b, h, w\}$, in which a, b, w and h, (a, b) is the center point coordinates of the box, w and h are its width and height.

For the classification loss, the ARM and the ODM use the binary cross entropy loss (Shaheed et al., 2022) and multi-classification cross entropy loss, respectively.

For the tth sample, its label is $\boldsymbol{y}_t{}'$, and its predicted output is $\boldsymbol{y}_t$, then the binary cross entropy loss is:

$$L_b(\boldsymbol{y}_t, \boldsymbol{y}_t') = -\sum_{i=t}^{N_{ARM}} (\boldsymbol{y}_t log(\boldsymbol{y}_t') + (1 - \boldsymbol{y}_t) log(1 - \boldsymbol{y}_t')) \tag{21}$$

where N_{ARM} is the number of training samples.

For the label of the cth category in the tth sample is $\boldsymbol{y}_{t,c}'$, the multi-classification cross entropy loss is:

$$L_m(\boldsymbol{y}_{t,c}, \boldsymbol{y}_{t,c}') = -\sum_{t=1}^{N_{ODM}} \sum_{c=1}^{C} \boldsymbol{y}_{t,c} log \boldsymbol{y}_{t,c}' \tag{22}$$

where N_{ODM} is the number of training samples and C is the number of categories.

The loss functions of the anchor refinement module and target detection module can be expressed as:

$$L_{ARM}(\boldsymbol{y}_t, \boldsymbol{y}_t', p^u, v) = L_b(\boldsymbol{y}_t, \boldsymbol{y}_t') + L_r(p^u, v) \tag{23}$$

$$L_{ODM}(\boldsymbol{y}_{t,c}, \boldsymbol{y}_{t,c}', p^u, v) = L_m(\boldsymbol{y}_{t,c}, \boldsymbol{y}_{t,c}') + L_r(p^u, v) \tag{24}$$

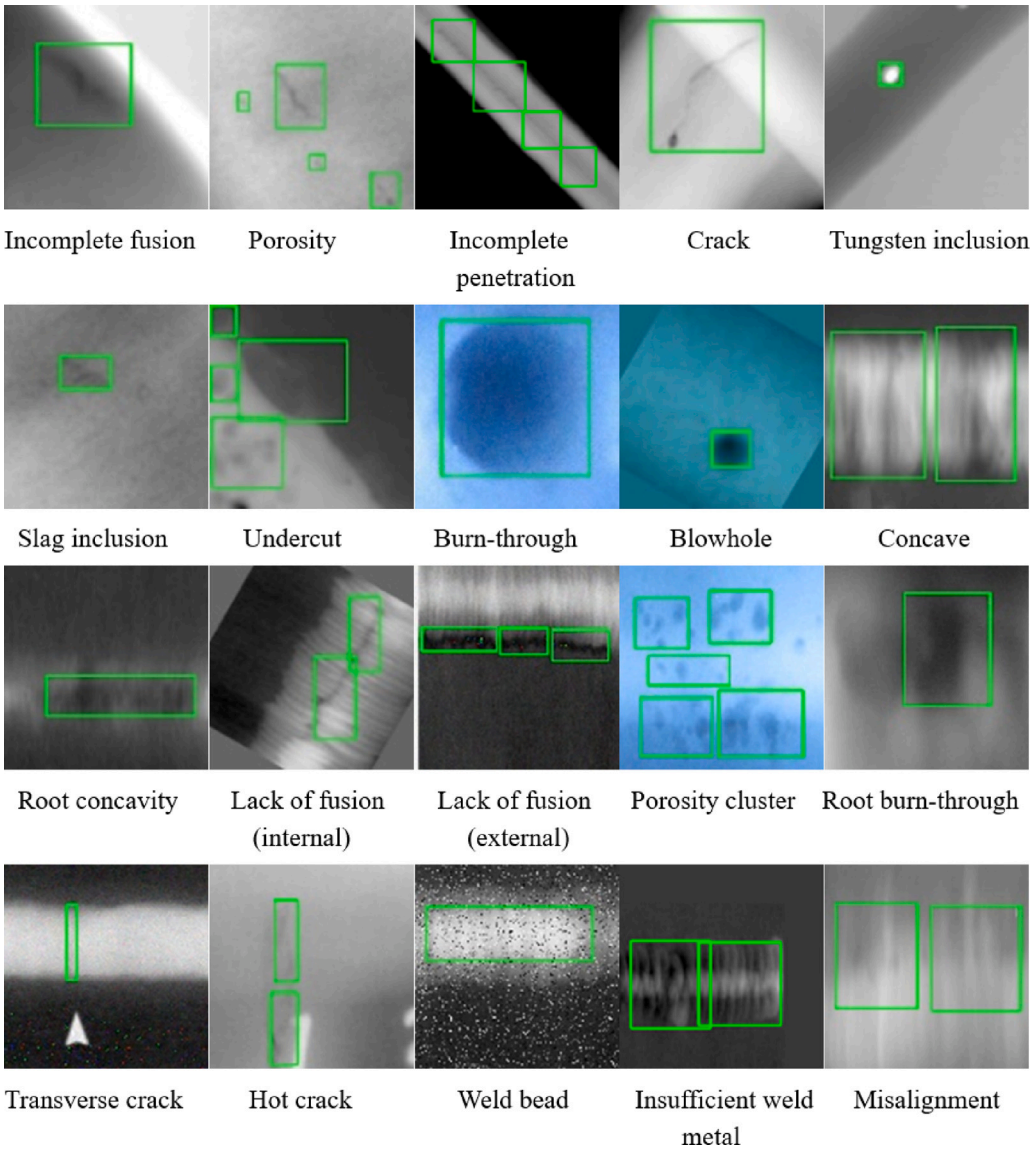


Fig. 9. Examples of the PIP-DET dataset.

4. Experiments

In this section, we compare the detection performance of Im-RefineDet with other methods to evaluate its effectiveness. This section explains in turn the experimental dataset, evaluation metrics, training details, ablation experiments, and comparison of experimental results with other methods, as well as providing corresponding analysis. The deep learning framework used in all experiments in this paper is Pytorch with the version of 1.11.0, and the version of Python is 3.8.10. We train our model with a single NVIDIA RTX 3090 GPU card.

4.1. Datasets

We constructed a pipeline defect dataset (PIP-DET) for experiments. It is a gas pipeline defect image dataset, which contains 20 defect categories and a total of 6010 samples. We use Labelimg (Tzutalin, 2022) to obtain the corresponding image labels. With a ratio of 7:3, the dataset is divided into two parts: the training dataset and the test dataset. Accordingly the training dataset and the test dataset have 4258 and 1752 images, respectively. During training and testing, the image size in the dataset is uniformly adjusted to 320 × 320. Table 1 provides the details of PIP-DET. Fig. 9 shows examples of different defect image annotations.

4.2. Evaluation metrics

The evaluation metrics used in this paper are mean Average Precision (mAP) (Lin et al., 2014), Params and FLOPs.

mAP refers to the average precision of all categories, which is calculated as:

$$\text{Precision} = \frac{\text{Tp}}{\text{Tp} + \text{Fp}} \tag{25}$$

$$\text{Recall} = \frac{\text{Tp}}{\text{Tp} + \text{Fn}} \tag{26}$$

$$\text{AP}_i = \sum_{j=0}^{n-1} (R_{j+1} - R_j) p_j^* \tag{27}$$

$$\text{mAP} = \frac{1}{N_c} \sum_{i=1}^{N_c} \text{AP}_i \tag{28}$$

where Tp, Fp and Fn denote the number of true positives, false positives and false negatives samples, respectively; n means the number of times

Table 1
The detail information of PIP-DET.

No.	Class	Samples	No.	Class	Samples
1	Incomplete fusion	694	11	Root concavity	417
2	Porosity	697	12	Lack of fusion (internal)	217
3	Incomplete penetration	699	13	Lack of fusion (external)	327
4	Crack	389	14	Porosity cluster	243
5	Tungsten inclusion	699	15	Root burn-through	385
6	Slag inclusion	710	16	Transverse crack	340
7	Undercut	286	17	Hot crack	271
8	Burn-through	364	18	Weld bead	482
9	Blowhole	417	19	Insufficient weld metal	300
10	Concave	283	20	Misalignment	392

Table 2
Detection accuracies with different improved methods.

No.	SCCEL	V-H-SWSA	DADH	mAP	mAP_{50}	mAP_{75}	mAP_s	mAP_m	mAP_l
1	✗	✗	✗	55.8	92.1	61.3	32.6	48.7	61.2
2	✔	✗	✗	56.4	92.4	61.4	32.5	48.6	62.3
3	✗	✔	✗	56.7	92.1	63.6	31.9	48.8	62.5
4	✗	✗	✔	56.9	92.4	61.7	32.5	48.8	63.3
5	✔	✔	✗	57.1	92.5	63.0	30.8	47.8	**63.9**
6	✗	✔	✔	57.1	92.2	63.2	32.4	49.2	62.9
7	✔	✗	✔	56.9	92.3	62.3	31.7	49.5	63.0
8	✔	✔	✔	**57.4**	**92.6**	**63.6**	**34.7**	**49.8**	63.0

to interpolate the precision–recall curve; the recall after interpolation is $R_j = \frac{j}{n}$, $j \in \{0, 1, 2, \ldots, n\}$; p_j^* represents precision at R_j; N_c is the number of categories; AP_i represents the average accuracy of the ith class.

Different metrics are used to measure model performance, including the overall mean Average Precision (mAP), which is the average value across all categories under the Intersection over Union (IoU) threshold of $[0.5, 0.95]$ with a step size of 0.05.

Additionally, the mAP at an IoU threshold of 0.50 is referred to as mAP_{50}, while the mAP at an IoU threshold of 0.75 is denoted as mAP_{75}. The metric mAP_s specifically measures the performance on small objects, defined as those with a pixel area size of $area < 32^2$. For medium-sized objects, indicated by the metric mAP_m, the pixel area size falls within the range $32^2 < area < 96^2$.

Finally, the metric mAP_l refers to large objects, which are defined as having a pixel area size greater than 96^2.

Params and FLOPs represent the parameter amount and calculation amount of the model respectively, and the units are M (size of 1×10^6) and G (size of 1×10^9), respectively.

4.3. Implementation protocol

The number of blocks in the four stages of the backbone network is 3, 3, 9, and 3, with attention head counts of 2, 4, 8, and 16. The drop path rate is set to 0.2, and the MLP ratio is 2. In the V-SWSA and H-SWSA modules, the attention window sizes are set to 5×10 and 10×5, respectively, and the channel pruning rate of the model is 30%.

During training, the initial learning rate is 5×10^{-4}. We employ the Adam (Kingma & Ba, 2015) to optimize, with the weight decay of 0.05 and the momentum of 0.9. Warmup (Goyal et al., 2017) and Cosine (Loshchilov & Hutter, 2016) are used to adjust the strategy. The "Warmup" phase is used at the start of training to gradually increase the learning rate from a very small value to the initial target learning rate over a few epochs. This helps to stabilize the training process and prevent large updates to the model parameters in the early stages, which can be especially important when using high learning rates. The "Cosine" learning rate strategy adjusts the learning rate following a cosine curve throughout the training process. At the beginning of training, the learning rate remains relatively high, and then it gradually decays towards zero as training progresses. This slow decay ensures a steady learning process, allowing the network to converge more slowly but often with better final accuracy.

In detail, the learning rate increases with a linear pattern from 0 to 5×10^{-4} in the first 10 epochs of training. Additionally, the remaining training times learning rate gradually decreases from 5×10^{-4} to 1×10^{-6} in the form of cosine function.

Additionally, the training epochs is 130, the batch size is 14. We use the SSD's data augmentation method to increase the number of training samples (Liu et al., 2016), including coordinate conversion, pixel content transformation, spatial geometry transformation, coordinate transformation and scaling, and mean subtraction.

4.4. Ablation study

4.4.1. Ablation study of the different improved methods

To evaluate the contributions of each enhancement to the overall model performance, we provides a clear understanding of how each component influences the model's success. We use different improved methods to observe the effectiveness of each method.

Table 2 shows the corresponding detection results. Among them, H-V-SWSA is vertical–horizontal SWSA. The baseline model is the RefineDet model that uses CrossFormer as the backbone network (Zhang et al., 2018).

From Table 2, it can be observed that: Firstly, all improvement methods (No. 2 to No. 8) show better results compared to the baseline model (No. 1). Specifically, mAP, mAP_{50}, and mAP_{75} increased by up to 1.6%, 0.5%, and 2.3%, respectively, while mAP_s, mAP_m and mAP_l improved by up to 2.1%, 1.1%, and 2.7%, respectively. This demonstrates that the improved backbone network, window self-attention mechanism, and detection head can effectively enhance RefineDet's detection capabilities. Secondly, comparing the results using all three improved methods (No. 8) with those using only one improved method (No. 2, No. 3, and No. 4), we observe improvements in all metrics except mAP_l. Notably, mAP, mAP_{50} and mAP_{75} improved by up to 1.0%, 0.5%, and 2.2% respectively, while mAP_s and mAP_m increased by up to 2.8% and 1.2% respectively. This indicates that simultaneously employing the improved backbone network, window self-attention mechanism, and detection head yields better detection performance than using just one improvement. Furthermore, when comparing the results of using all three improved methods (No. 8) with those using two improved methods (No. 5, No. 6, and No. 7), we again see improvements across

Table 3
Accuracy of CrossFormer using different types of window self-attention.

No.	Windows self-attention type	mAP	mAP_{50}	mAP_{75}	mAP_s	mAP_m	mAP_l
1	WSA(Baseline)	56.2	91.6	62.1	31.1	47.6	61.6
2	A-WSA	56.6	92.1	62.6	32.5	48.7	62.1
3	C-SWSA	56.8	92.2	62.8	32.6	48.4	62.4
4	H-SWSA	56.4	92.0	62.7	32.1	47.9	62.0
5	V-SWSA	56.5	91.7	62.6	32.3	48.2	62.3
6	V-H-SWSA(Ours)	**57.0**	**92.3**	**63.0**	**33.4**	**48.6**	**62.5**

Table 4
Comparison with state-of-the-art methods on the PIP-DET dataset.

Style	Methods	Backbone	mAP	mAP_{50}	mAP_{75}	mAP_s	mAP_m	mAP_l	Params	FLOPs	FPS
Two-stage	Faster RCNN	ResNet50	52.6	89.2	54.1	20.4	42.1	59.6	41.45M	20.37G	56.0
	Cascade RCNN	ResNet50	53.3	89.0	56.3	18.2	43.7	60.4	69.21 M	22.37G	54.7
One-stage	SSD	VGG16	50.2	88.0	51.8	26.4	39.7	56.0	26.29M	35.31G	50.6
	YOLOv3	Darknet53	50.8	88.8	52.5	26.1	43.0	56.6	61.50M	19.40G	58.5
	YOLOv5	CSPDarknet53	57.2	92.0	62.0	**36.4**	48.0	63.5	46.24M	13.53G	66.5
	FINet	CSPDarknet53	54.0	90.6	48.2	25.8	38.7	54.1	**7.30M**	**4.30G**	**78.5**
	DEA-RetinaNet	ResNet50	51.6	87.5	52.9	19.9	40.4	59.6	41.03M	22.86G	56.1
Transformer	Deformable DETR	ResNet50	54.9	88.9	59.1	29.1	45.7	60.9	41.05M	23.47G	54.3
	DAB DETR	ResNet50	56.9	91.7	61.3	34.2	46.4	63.6	47.42M	31.93G	52.8
	DN DETR	ResNet50	57.1	91.6	63.1	35.9	48.0	**63.9**	47.30M	31.47G	52.6
Mixed	RefineDet	VGG16	54.2	91.5	58.2	28.0	46.3	59.4	36.16M	140.67G	20.4
	Im-RefineDet	Im-CrossFormer	**57.4**	**92.6**	**63.6**	34.7	**49.8**	63.0	56.31M	72.10G	28.6
	Im-RefineDet-lite	Im-CrossFormer-lite	57.2	92.3	63.1	34.3	49.5	62.6	22.57M	38.14G	47.5

all metrics except mAP_l. Specifically, mAP, mAP_{50} and mAP_{75} improved by up to 0.5%, 0.4%, and 1.3% respectively, while mAP_s and mAP_m increased by up to 3.9% and 2% respectively. This further confirms that utilizing all three improved methods simultaneously can achieve higher detection accuracy. Finally, by incorporating all three improved methods (SCCEL, SWSA, and DADH), the model achieved the best results in mAP, mAP_{50}, mAP_{75}, mAP_s and mAP_m, reaching 57.4%, 92.6%, 63.6%, 34.7%, and 49.8% respectively. This conclusively demonstrates that the simultaneous application of these three improved methods contributes to obtaining superior detection results.

4.4.2. Ablation study of the SWSA

To explore the impact of different window self-attention mechanisms on detection performance in the pipeline defect dataset, we seek to determine how the directional characteristics of horizontal and vertical attention windows contribute to improved feature capture and overall detection accuracy.

Table 3 presents the detection results of CrossFormer using various types of window self-attention on the pipeline defect dataset. Here, WSA indicates the standard window self-attention with equal width and height (Liu et al., 2021). A-WSA is Axial Windows Self-Attention (Wang et al., 2020), while C-SWSA is Cross-Shaped Windows Self-Attention (Dong et al., 2022). H-SWSA and V-SWSA represent Horizontal Stripes Windows Self-Attention and Vertical Stripes Windows Self-Attention, respectively. V-H-SWSA refers to the simultaneous use of both Vertical and Horizontal Windows Self-Attention.

From Table 3, it can be observed that: Compared to the results of WSA (No. 1), the results of using V-H-SWSA (No. 6) show improvements in mAP, mAP_{50}, mAP_{75}, mAP_s, mAP_m, and mAP_l, with increases of 0.8%, 0.7%, 0.9%, 2.3%, 1.0%, and 0.9%, respectively. Additionally, compared to the results of A-WSA and C-SWSA (No. 2 and No. 3), the results of using V-H-SWSA (No. 6) show improvements in mAP, mAP_{50}, mAP_{75}, mAP_s, mAP_m, and mAP_l, with the highest increases of 0.4%, 0.2%, 0.4%, 0.9%, 1.2%, and 0.4%, respectively. Moreover, compared to the results of H-WSA and V-SWSA (No. 4 and No. 5), the results of using V-H-SWSA (No. 6) show improvements in mAP, mAP_{50}, mAP_{75}, mAP_s, mAP_m, and mAP_l, with the highest increases of 0.5%, 0.6%, 0.4%, 1.3%, 0.7% and 0.5%, respectively. This progressive improvement indicates that the bidirectional focus of V-H-SWSA provides a broader context for feature extraction, leading to more robust detection capabilities. In summary, by simultaneously utilizing both vertical and horizontal attention, V-H-SWSA enables a more comprehensive feature extraction process, enhancing the overall detection performance.

4.5. Comparison Im-RefineDet with different methods on pipeline defect dataset

To evaluate the performance of the Im-RefineDet, we compare it with the state-of-the-art detection methods, as well as providing the corresponding analysis. Table 4 displays the detection results on the dataset PIP-DET, where the bold numbers denote the best results.

From Table 4, it can be observed that: when compared to two-stage methods, Im-RefineDet demonstrates significant improvements in detection performance. It enhances mAP, mAP_{50}, and mAP_{75} by up to 4.8%, 7.7%, and 9.5%, respectively. For targets of different sizes, mAP_s, mAP_m, and mAP_l show improvements of up to 16.5%, 3.6%, and 3.4%, respectively. Additionally, in comparison with single-stage models, Im-RefineDet shows remarkable improvements. It enhances mAP, mAP_{50}, and mAP_{75} by up to 7.2%, 5.1%, and 11.4%, respectively. Particularly in detecting medium-sized targets, Im-RefineDet's mAP_m surpasses these five single-stage models by 10.1%, 6.8%, 1.8%, 11.1%, and 9.4%, respectively. Besides, when compared to the hybrid model RefineDet, Im-RefineDet shows notable improvements. It enhances mAP, mAP_{50}, and mAP_{75} by 3.2%, 1.1%, and 5.4%, respectively. For targets of different sizes, mAP_s, mAP_m, and mAP_l increase by 6.7%, 3.5%, and 3.6%, respectively. Moreover, Im-RefineDet also outperforms detection Transformer models, improving mAP, mAP_{50}, and mAP_{75} by 2.3%, 3.4%, and 4.0%, respectively. For targets of different sizes, mAP_s, mAP_m, and mAP_l increase by 5.2%, 3.8%, and 1.7%, respectively. This indicates that Im-RefineDet achieves higher accuracy than state-of-the-art Transformer-based detection models.

In detecting small objects, Im-RefineDet achieves the second-best mAP_s at 34.7%. This is attributed to the addition of an attention module, which allows the model to focus more on critical areas of the image. However, small targets, occupying fewer pixels, still pose challenges as their feature information can be easily overlooked, making it difficult for the model to learn accurate feature representations.

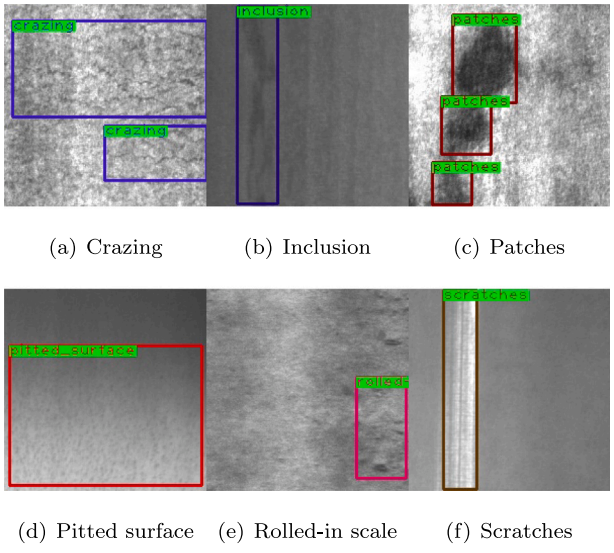
In conclusion, compared to existing mainstream single-stage and two-stage detection models, Im-RefineDet achieves optimal overall mAP performance.

Table 5
Comparison with state-of-the-art methods on the NEU-DET dataset.

Style	Methods	Backbone	mAP_{50}	Crazing	Inclusion	Patches	Pitted surface	Rolled-in scale	Scratches
Two-stage	Faster RCNN	ResNet50	71.2	36.3	72.8	87.8	86.7	56.1	87.5
	Cascade RCNN	ResNet50	74.9	46.5	76.8	88.7	87.0	60.8	89.7
One-stage	SSD	VGG16	66.5	31.4	66.1	86.1	78.6	**65.7**	71.2
	YOLOv3	Darknet53	71.3	30.1	75.3	89.0	83.8	55.8	93.7
	YOLOv5	CSPDarknet53	77.3	46.7	78.3	**91.1**	87.0	65.4	**94.8**
	FINET	CSPDarknet53	71.4	37.1	78.9	91.0	83.0	57.3	81.0
	DEA-RetinaNet	ResNet50	70.6	36.4	74.2	90.0	86.5	63.7	72.4
Transformer	Deformable DETR	ResNet50	75.4	46.8	78.4	89.7	85.5	61.1	90.7
	DAB DETR	ResNet50	75.2	45.4	72.7	88.1	85.2	62.8	93.8
	DN DETR	ResNet50	75.7	47.7	78.7	89.9	82.2	62.9	92.9
Mixed	RefineDet	VGG16	76.1	47.3	**80.2**	90.5	85.4	60.0	93.1
	Im-RefineDet	Im-CrossFormer	**77.8**	**51.6**	79.6	89.4	**88.8**	64.0	93.1
	Im-RefineDet-lite	Im-CrossFormer-lite	77.5	51.4	79.4	89.2	88.5	63.7	93.0

Table 6
Comparison with state-of-the-art methods on the PCB dataset.

Methods	mAP_{50}	Missing hole	Mouse bite	Open circuit	Short	Spur	Spurious copper
Faster RCNN	92.2	92.5	92.3	91.5	92.4	92.6	91.9
Cascade RCNN	93.1	93.4	93.5	93.0	93.2	92.8	92.7
SSD	87.5	87.3	87.6	87.5	88.1	87.2	87.4
YOLOv3	94.9	95.2	95.0	94.5	95.4	94.6	94.9
YOLOv5	98.5	98.7	97.8	98.5	98.6	98.9	98.4
FINet	92.8	92.4	92.5	93.0	93.2	92.8	92.7
DEA-RetinaNet	92.3	92.6	92.3	91.7	92.4	92.5	92.0
Deformable DETR	93.2	93.2	93.4	93.3	93.4	93.0	93.1
DAB DETR	97.5	97.8	97.5	97.2	97.4	97.8	97.5
DN DETR	97.8	97.9	97.8	97.4	97.7	97.1	97.6
RefineDet	95.3	95.6	95.4	94.9	95.0	95.5	95.1
Im-RfineDet	**99.2**	**99.3**	**99.1**	**99.5**	**99.2**	**99.0**	**99.1**
Im-RfineDet-lite	98.9	99.0	98.9	99.2	98.7	98.6	98.8



(a) Crazing (b) Inclusion (c) Patches

(d) Pitted surface (e) Rolled-in scale (f) Scratches

Fig. 10. Examples of defect images with annotations in NEU-DET.

4.6. Comparison Im-RefineDet with different methods on public dataset NEU-DET

To verify the performance of the Im-RefineDet on public dataset, we select a public dataset NEU-DET to do experiments.

NEU-DET is a steel surface defect dataset (He, Song, Meng, & Yan, 2019). This dataset contains 6 defect categories, with 300 samples for each category. There are a total of 1800 samples and corresponding labels. Further, the training dataset and the test dataset have 1260 and 540 samples, respectively. Fig. 10 shows an example of defect image annotation.

Table 7
Comparison of computational efficiency across different models.

Method	mAP_{50}	Params	FLOPs	FPS
Faster RCNN	89.2	41.45M	20.37G	56.0
Cascade-RCNN	89.0	69.21 M	22.37G	54.7
SSD	88.0	26.29M	35.31G	50.6
YOLOv3	88.8	61.50M	19.40G	58.5
YOLOv5	92.0	46.24M	13.53G	66.5
FINet	90.6	**7.30M**	**4.30G**	**78.5**
DEA-RetinaNet	87.5	41.03M	22.86G	56.1
Deformable DETR	88.9	41.05	23.47	54.3
DAB DETR	91.7	47.42	31.93	52.8
DN DETR	91.6	47.30	31.47	52.6
RefineDet	91.5	36.16M	140.67G	20.4
Im-RefineDet	**92.6**	56.31M	72.10G	28.6
Im-RefineDet-lite	92.3	22.57M	38.14G	47.5

The detail detection results of different methods are provided in Table 5, with the bold numbers indicating the best results.

From Table 5, it can be observed that: when compared to two-stage methods of Faster RCNN and Cascade RCNN, Im-RefineDet's mAP_{50} has increased by 6.6% and 2.9%, respectively. In the detection of Crazing category, mAP_{50} increased by 15.3% and 5.1%. In detecting the Pitted surface category, mAP_{50} improved by 2.1% and 1.8%. In other categories, it also has certain improvements. This demonstrates that it can achieve higher results than the existing mainstream two-stage models on this public dataset. Additionally, in comparison with single-stage models of SSD, YOLOv3, YOLOv5, FINet and DEA-RetinaNet, Im-RefineDet improves mAP_{50} by 11.3%, 6.5%, 0.5%, 6.4% and 7.2%, respectively. Especially in the Crazing category, Im-RefineDet's mAP_{50} is significantly higher than the above-mentioned single-stage model. This clarifies that the proposed In-RefineDet performs better than the state-of-the-art single-stage models on the NEU-DET dataset. Moreover, when compared to the hybrid model of RefineDet, Im-RefineDet improves mAP_{50} by 1.7%. In detecting of the types of defects, crazing,

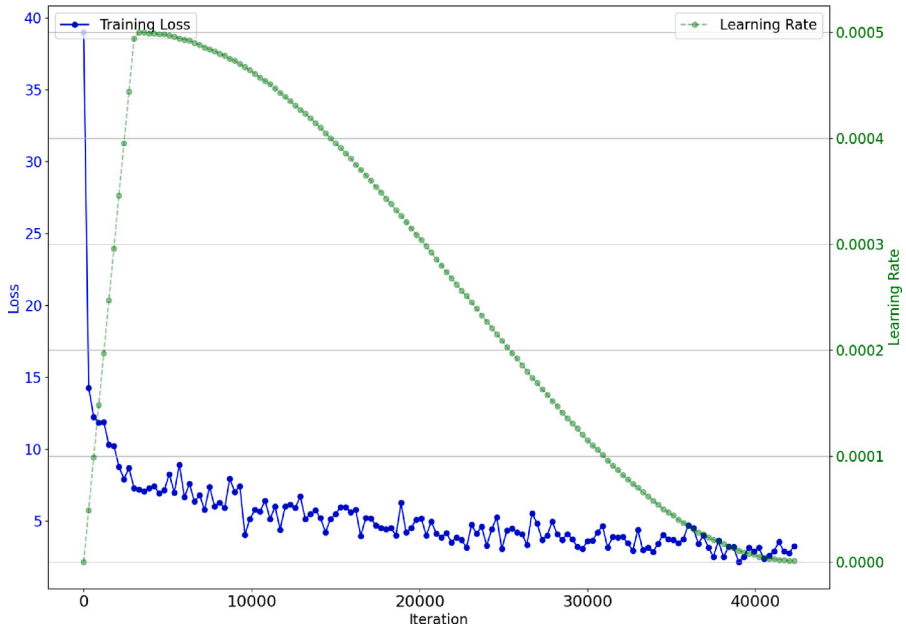


Fig. 11. Training Loss and learning rate at iterations (multiples of 300).

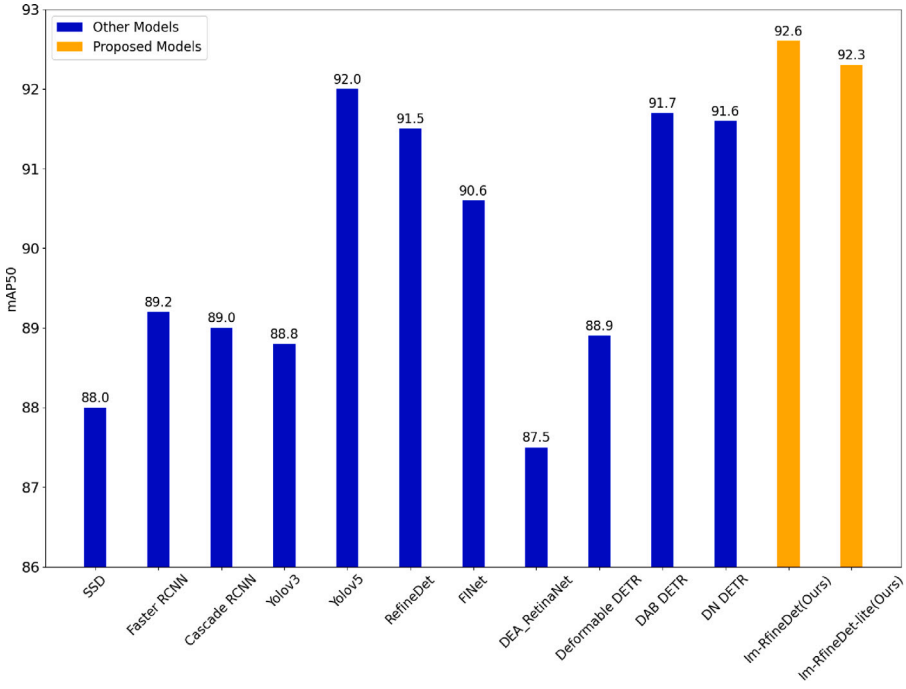


Fig. 12. The mAP_{50} results of different methods on the PIP-DET dataset.

Pitted surface and Rolled-in scale, mAP_{50} increased by 4.3%, 3.4% and 4.0%, respectively. Furthermore, Im-RefineDet surpasses detection Transformer models, improving mAP_{50} by 2.6%, with notable enhancements of 6.2% and 6.6% in the Crazing and Pitted surface categories respectively. It demonstrates that Im-RefineDet achieves higher accuracy than state-of-the-art Transformer-based detection models on the NEU-DET dataset.

In conclusion, Im-RefineDet achieved higher mAP_{50} than existing models on the public dataset. Besides, its detection accuracy in different categories has also been improved to varying degrees. This manifests that the proposed Im-RefineDet has certain generalization ability.

4.7. Comparison Im-RefineDet with different methods on PCB defect dataset

We have conducted further validation of our proposed method, Im-RefineDet, on the publicly available PCB defect dataset (Ding, Dai, Li, & Liu, 2019). This dataset is specifically designed for defect detection in printed circuit boards and consists of a total of 12,428 images across six defect categories: Missing hole, Mouse bite, Open circuit, Short, Spur, and Spurious copper.

The detail detection results of different methods are provided in Table 6. From Table 6, it can be concluded that: Im-RefineDet outperforms all other models across the board, achieving a mAP_{50} of 99.2%.

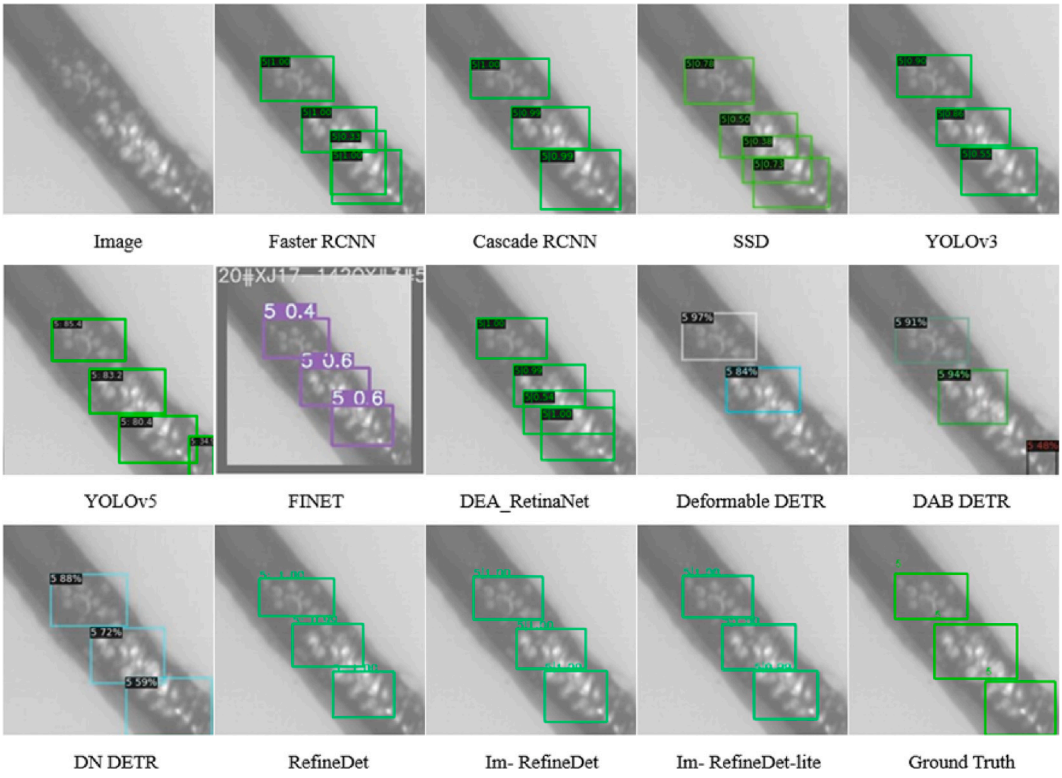


Fig. 13. Detection examples of different methods on PIP-DET dataset.

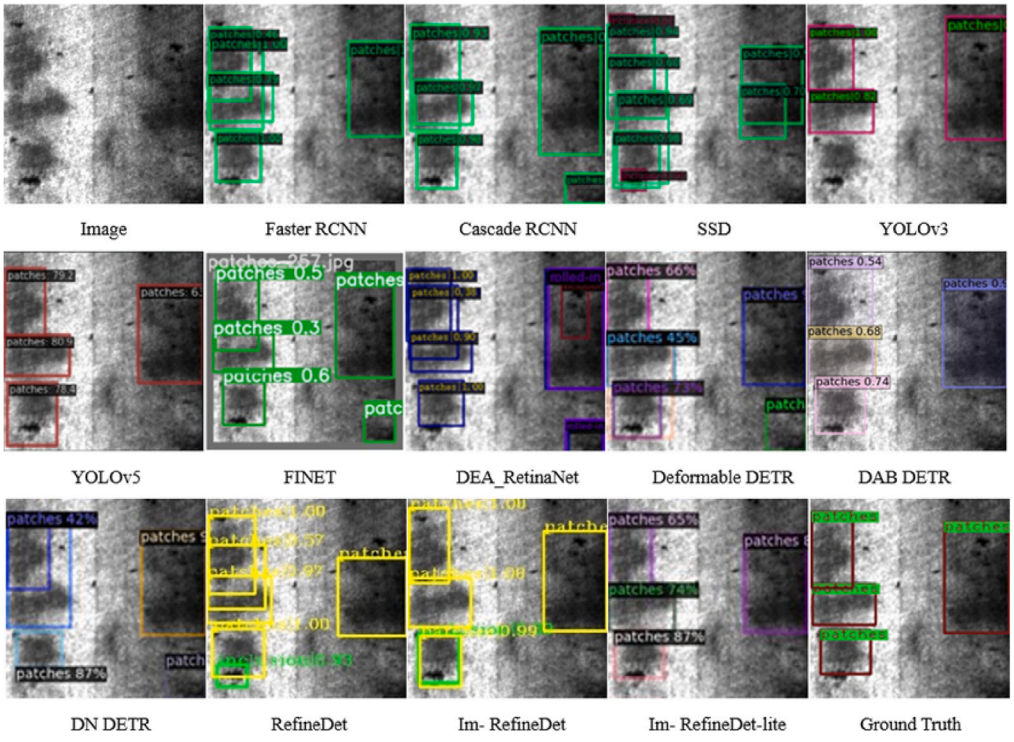


Fig. 14. Detection examples of different methods on NEU-DET dataset.

This demonstrates its effectiveness in detecting defects in printed circuit boards. The slight drop in performance for Im-RefineDet-lite indicates that channel pruning can effectively reduce model complexity without significantly compromising accuracy, making it suitable for resource-constrained environments.

4.8. Comparison of computational efficiency

This section provides a detailed comparison of computational efficiency across different models, including Params(parameters), FLOPs (floating-point operations) and FPS (frames per second) of different models, and gives their performance in Table 7.

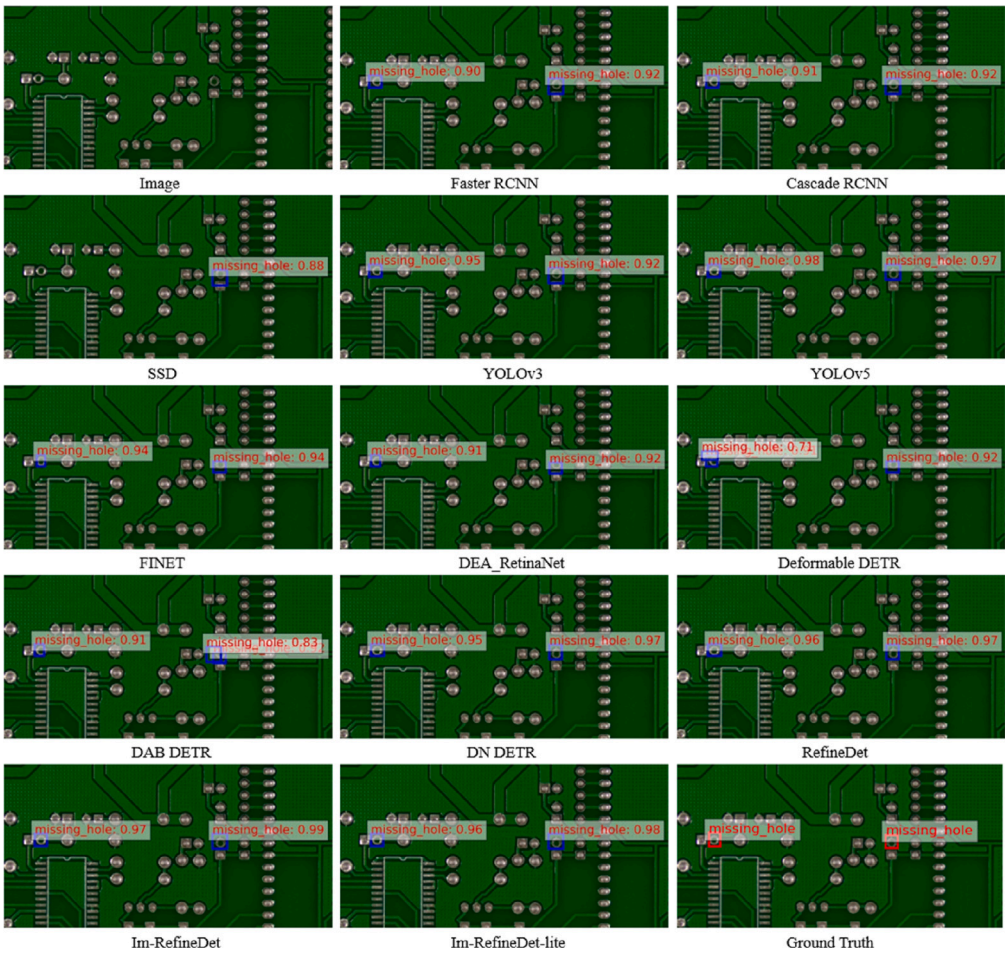


Fig. 15. Detection examples of different methods on PCB dataset.

From Table 7, it can be observed that: For the Parameters, FINet (7.30M) and Im-RefineDet-lite (22.57M) have the smallest parameter counts, making them suitable for resource-limited devices. Cascade-RCNN (69.21M) and Im-RefineDet (56.31M) have the largest, which may lead to higher memory and storage requirements. For the FLOPs, FINet (4.30G) and YOLOv5 (13.53G) have the lowest FLOPs, which makes them highly efficient, especially for real-time detection. RefineDet (140.67G) has higher FLOPs than the other models, leading to slower inference speeds and unsuitability for real-time applications. With channel pruning, Im-RefineDet-lite reduces its FLOPs from 72.10G to 38.14G, nearly halving its computational load and significantly improving efficiency. For the FPS, FINet (78.5 FPS) achieves the best performance in real-time scenarios due to its low parameter count and FLOPs. Im-RefineDet-lite boosts its FPS from 28.6 to 47.5 after pruning, greatly improving inference speed along with reduced computation.

Additionally, compared to Im-RefineDet, by applying the channel pruning method, Im-RefineDet-lite's FLOPs reduced from 72.10G to 38.14G, and the number of parameters decreased from 55.31M to 22.57M, while its mAP_{50} decreased from 92.6% to 92.3%. Our method achieves an inference speed of 47.5 FPS on a 4060 GPU, indicating its potential suitability for real-time applications. It can be concluded that, despite the significant reduction in computational and parameter counts, the accuracy only decreased by 0.2% and 0.3%.

4.9. Visualization analysis

(1) Visualization of training and testing processes

The results of Fig. 11 demonstrate a continuous decline in training loss as the iterations progress, eventually stabilizing, which indicates successful model convergence. The learning rate adjustment strategy, starting high to accelerate convergence and decreasing later, further optimizes model performance.

Fig. 12 shows that the proposed models, Im-RefineDet and Im-RefineDet-lite, achieve mAP_{50} scores of 92.6% and 92.3%, respectively, significantly surpassing other mainstream models. These results collectively validate the training efficiency and detection accuracy of the proposed models.

(2) Visualization of detection results

In Fig. 13, the PIP-DET results show that the detection boxes accurately cover all major areas of the targets, with high confidence scores, reducing redundant boxes and improving clarity compared to other methods. Similarly, in Fig. 14, the NEU-DET results indicate that the proposed method minimizes missed detections and provides high accuracy in identifying most targets, even though some instances of multiple detections occur in complex backgrounds. Lastly, in Fig. 15, the PCB results highlight the method's ability to detect small targets with high confidence, accurately predicting positions and addressing issues related to complex backgrounds, making it highly robust and applicable for such tasks. The analysis of the PIP-DET, NEU-DET and PCB datasets demonstrates the superior performance of the proposed method.

5. Conclusion

In this paper, we introduce an improved RefineDet model for pipeline defect detection, aimed at enhancing feature extraction and

detection performance through specific modifications to the backbone network and detection head. The backbone utilizes an improved CrossFormer with the SCCEL module to decompose large convolutional kernels into smaller, stride-2 kernels, reducing feature loss and expanding the receptive field for richer multi-scale feature capture. We also propose the Stripes Windows Self-Attention module, which accommodates the directional characteristics of defects by incorporating horizontal and vertical attention windows, thereby improving token interaction range without significantly increasing computational costs. Additionally, the model employs a double attention decoupled head to separate classification and regression tasks, allowing for targeted optimizations, and incorporates spatial and scale-aware attention to enhance feature fusion across scales. Collectively, these enhancements lead to superior accuracy and computational efficiency, as demonstrated on both public and custom datasets.

CRediT authorship contribution statement

Ting Zhang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Cong Ma:** Visualization, Validation, Software, Methodology, Investigation, Conceptualization, Writing – review & editing. **Zhaoying Liu:** Visualization, Validation, Software, Methodology, Investigation, Supervision, Resources. **Sadaqat ur Rehman:** Methodology, Conceptualization, Validation, Software, Writing – review & editing. **Yujian Li:** Writing – review & feedback. **Mohamad Saraee:** Writing – review & feedback.

Declaration of competing interest

The authors declare no conflict of interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61906005, 61806013, 61876010, 62166002, 62176009), Natural Science Foundation of Xinjiang Uygur Autonomous Region (2023D01A22). General Project of Science and Technology Plan of Beijing Municipal Education Commission (KM202110005028), and Foundation Project of the Science.

Data availability

Data will be made available on request.

References

Arya, A. K., Jain, R., Yadav, S., Bisht, S., & Gautam, S. (2022). Recent trends in gas pipeline optimization. *Materials Today: Proceedings*, *57*, 1455–1461. http://dx.doi.org/10.1016/j.matpr.2021.11.232.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. http://dx.doi.org/10.48550/arXiv.2004.10934, arXiv preprint arXiv:2004.10934.

Cai, Z., & Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6154–6162). http://dx.doi.org/10.1109/cvpr.2018.00644.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision* (pp. 213–229). Springer, http://dx.doi.org/10.48550/arXiv.2005.12872.

Chen, M., Yu, L., Zhi, C., Sun, R., Zhu, S., Gao, Z., et al. (2022). Improved faster R-CNN for fabric defect detection based on gabor filter with genetic algorithm optimization. *Computers in Industry*, *134*, Article 103551. http://dx.doi.org/10.1016/j.compind.2021.103551.

Cheng, X., & Yu, J. (2020). RetinaNet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–11. http://dx.doi.org/10.1109/tim.2020.3040485.

Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., et al. (2021). Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7373–7382). http://dx.doi.org/10.1109/cvpr46437.2021.00729.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. http://dx.doi.org/10.48550/arXiv.1810.04805, arXiv preprint arXiv:1810.04805.

Ding, R., Dai, L., Li, G., & Liu, H. (2019). TDD-net: a tiny defect detection network for printed circuit boards. *CAAI Transactions on Intelligence Technology*, *4*(2), 110–116. http://dx.doi.org/10.1049/trit.2019.0019.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., et al. (2022). Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 12124–12134). http://dx.doi.org/10.1109/cvpr52688.2022.01181.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International conference on learning representations*. http://dx.doi.org/10.48550/arXiv.2010.11929.

Gao, L., Zhang, J., Yang, C., & Zhou, Y. (2022). Cas-vswin transformer: A variant swin transformer for surface-defect detection. *Computers in Industry*, *140*, Article 103689. http://dx.doi.org/10.1016/j.compind.2022.103689.

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). Yolox: Exceeding yolo series in 2021. http://dx.doi.org/10.48550/arXiv.2107.08430, arXiv preprint arXiv:2107.08430.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1440–1448). http://dx.doi.org/10.1109/iccv.2015.169.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., et al. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. http://dx.doi.org/10.48550/arXiv.1706.02677, arXiv preprint arXiv:1706.02677.

He, Y., Song, K., Meng, Q., & Yan, Y. (2019). An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Transactions on Instrumentation and Measurement*, *69*(4), 1493–1504. http://dx.doi.org/10.1109/tim.2019.2915404.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778). http://dx.doi.org/10.1109/cvpr.2016.90.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Michael, K., et al. (2022). ultralytics/yolov5: v7.0-yolov5 sota realtime instance segmentation. https://github.com/ultralytics/yolov5.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*. http://dx.doi.org/10.48550/arXiv.1412.6980.

Layouni, M., Hamdi, M. S., & Tahar, S. (2017). Detection and sizing of metal-loss defects in oil and gas pipelines using pattern-adapted wavelets and machine learning. *Applied Soft Computing*, *52*, 247–261. http://dx.doi.org/10.1016/j.asoc.2016.10.040.

Li, Y., Yao, T., Pan, Y., & Mei, T. (2022). Contextual transformer networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(2), 1489–1500. http://dx.doi.org/10.48550/arXiv.2107.12292.

Li, F., Zhang, H., Liu, S., Guo, J., Ni, L. M., & Zhang, L. (2022). Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13619–13627). http://dx.doi.org/10.1109/cvpr52688.2022.01325.

Li, W., Zhang, H., Wang, G., Xiong, G., Zhao, M., Li, G., et al. (2023). Deep learning based online metallic surface defect detection method for wire and arc additive manufacturing. *Robotics and Computer-Integrated Manufacturing*, *80*, Article 102470. http://dx.doi.org/10.1016/j.rcim.2022.102470.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755). http://dx.doi.org/10.48550/arXiv.1405.0312.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37). http://dx.doi.org/10.1007/978-3-319-46448-0.2.

Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., et al. (2022). Dab-detr: Dynamic anchor boxes are better queries for detr. http://dx.doi.org/10.48550/arXiv.2201.12329, arXiv preprint arXiv:2201.12329.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 10012–10022). http://dx.doi.org/10.1109/iccv48922.2021.00986.

Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. http://dx.doi.org/10.48550/arXiv.1608.03983, arXiv preprint arXiv:1608.03983.

Luo, J., Yang, Z., Li, S., & Wu, Y. (2021). FPCB surface defect detection: A decoupled two-stage object detection framework. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–11. http://dx.doi.org/10.1109/tim.2021.3092510.

Mezher, A. M., & Marble, A. E. (2024). Computer vision defect detection on unseen backgrounds for manufacturing inspection. *Expert Systems with Applications*, *243*, Article 122749. http://dx.doi.org/10.1016/j.eswa.2023.122749.

Mittal, P., Sharma, A., Singh, R., & Dhull, V. (2022). Dilated convolution based RCNN using feature fusion for low-altitude aerial objects. *Expert Systems with Applications*, *199*, Article 117106. http://dx.doi.org/10.1016/j.eswa.2022.117106.

Qi, S., Yang, J., & Zhong, Z. (2020). A review on industrial surface defect detection based on deep learning technology. In *Proceedings of the 2020 3rd international conference on machine learning and machine intelligence* (pp. 24–30). http://dx.doi.org/10.1145/3426826.3426832.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-alone self-attention in vision models. In *Advances in neural information processing systems*: *vol. 32*, http://dx.doi.org/10.48550/arXiv.1906.05909.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788). http://dx.doi.org/10.1109/cvpr.2016.91.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 6517–6525). http://dx.doi.org/10.1109/CVPR.2017.690.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. http://dx.doi.org/10.48550/arXiv.1804.02767, arXiv preprint arXiv:1804.02767.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*: *vol. 28*, http://dx.doi.org/10.1109/tpami.2016.2577031.

Shafi, I., Mazahir, A., Fatima, A., & Ashraf, I. (2022). Internal defects detection and classification in hollow cylindrical surfaces using single shot detection and MobileNet. *Measurement*, *202*, Article 111836. http://dx.doi.org/10.1016/j.measurement.2022.111836.

Shaheed, K., Mao, A., Qureshi, I., Kumar, M., Hussain, S., Ullah, I., et al. (2022). DS-CNN: A pre-trained xception model based on depth-wise separable convolutional neural network for finger vein recognition. *Expert Systems with Applications*, *191*, Article 116288. http://dx.doi.org/10.1016/j.eswa.2021.116288.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. http://dx.doi.org/10.1109/cvpr.2016.182, arXiv preprint arXiv:1409.1556.

Song, G., Liu, Y., & Wang, X. (2020). Revisiting the sibling head in object detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 11563–11572). http://dx.doi.org/10.1109/cvpr42600.2020.01158.

Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., et al. (2023). Sparse R-CNN: An end-to-end framework for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, http://dx.doi.org/10.1109/tpami.2023.3292030.

Tian, X., Jiao, W., Liu, T., Ren, L., & Song, B. (2022). Intelligent detection method of low-pressure gas system leakage based on semi-supervised anomaly diagnosis. *Expert Systems with Applications*, *209*, Article 118376. http://dx.doi.org/10.1016/j.eswa.2022.118376.

Tzutalin, D. (2022). LabelImg is a graphical image annotation tool and label object bounding boxes in images. Retrieved from https://github.com/tzutalin/labelImg.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems*: *vol. 30*, http://dx.doi.org/10.48550/arXiv.1706.03762.

Wang, W., Yao, L., Chen, L., Lin, B., Cai, D., He, X., et al. (2022). CrossFormer: A versatile vision transformer hinging on cross-scale attention. In *International conference on learning representations*. http://dx.doi.org/10.48550/arXiv.2108.00154.

Wang, H., Zhu, Y., Green, B., Adam, H., Yuille, A., & Chen, L.-C. (2020). Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European conference on computer vision* (pp. 108–126). http://dx.doi.org/10.48550/arXiv.1906.05909.

Wu, Y., Chen, Y., Yuan, L., Liu, Z., Wang, L., Li, H., et al. (2020). Rethinking classification and localization for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 10186–10195). http://dx.doi.org/10.1109/cvpr42600.2020.01020.

Wu, S., Wu, T., Tan, H., & Guo, G. (2022). Pale transformer: A general vision transformer backbone with pale-shaped attention. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2731–2739). http://dx.doi.org/10.1609/aaai.v36i3.20176.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Advances in neural information processing systems*: *vol. 34*, (pp. 12077–12090). http://dx.doi.org/10.48550/arXiv.2105.15203.

Yamamoto, K., & Maeno, K. (2018). Pcas: Pruning channels with attention statistics for deep network compression. http://dx.doi.org/10.48550/arXiv.1806.05382, arXiv preprint arXiv:1806.05382.

Yu, F., Huang, K., Wang, M., Cheng, Y., Chu, W., & Cui, L. (2022). Width & depth pruning for vision transformers. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 36*, (3), (pp. 3143–3151). http://dx.doi.org/10.1609/aaai.v36i3.20222.

Zeng, N., Wu, P., Wang, Z., Li, H., Liu, W., & Liu, X. (2022). A small-sized object detection oriented multi-scale feature fusion approach with application to defect detection. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1–14. http://dx.doi.org/10.1109/tim.2022.3153997.

Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2018). Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4203–4212). http://dx.doi.org/10.1109/cvpr.2018.00442.

Zhang, Z.-D., Zhang, B., Lan, Z.-C., Liu, H.-C., Li, D.-Y., Pei, L., et al. (2022). Finet: An insulator dataset and detection benchmark based on synthetic fog and improved YOLOv5. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1–8. http://dx.doi.org/10.1109/tim.2022.3194909.

Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., et al. (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6881–6890). http://dx.doi.org/10.1109/cvpr46437.2021.00681.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. In *International conference on learning representations*. http://dx.doi.org/10.48550/arXiv.2010.04159.