Hindawi
Journal of Sensors
Volume 2022, Article ID 3005684, 19 pages
https://doi.org/10.1155/2022/3005684

Research Article

LETR: An End-to-End Detector of Reconstruction Area in Blades Adaptive Machining with Transformer

Zikai Yin,[1] Yongshou Liang,[1] Junxue Ren,[1] Jungang An,[2] and Famei He[3]

[1]*Key Laboratory of High-Performance Manufacturing for Aero Engines, School of Mechanical Engineering, Northwestern Polytechnical University, 127 West Youyi Road, Beilin District, Xi'an 710072, China*
[2]*Haimo Research Institution, 22 Technology of Fifth Road, High Technology District, Xi'an 710000, China*
[3]*Beijing Institute of Technology, No. 5, South Street, Zhongguancun, Haidian District, Beijing 100000, China*

Correspondence should be addressed to Yongshou Liang; liangyongshou@nwpu.edu.cn

Received 23 September 2021; Accepted 8 April 2022; Published 14 May 2022

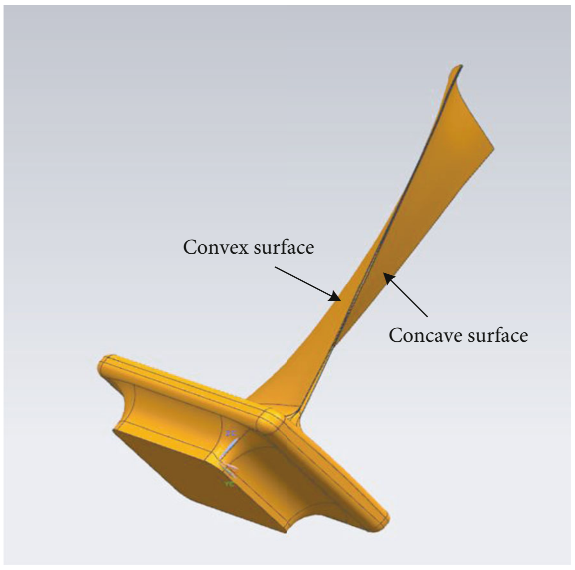
Academic Editor: Ehsan Namaziandost

In the leading/trailing edge's adaptive machining of the near-net-shaped blade, a small portion of the theoretical part, called the reconstruction area, is retained for securing aerodynamic performance by manual work. The next work is to recognize the reconstruction area of the reconstructed leading/trailing edge's image. To accelerate this process, an anchor-free neural network model based on Transformer was proposed, named Leading/trailing Edge Transformer (LETR). LETR extracts image features from an aspect of mixed frequency and channel domain. We also integrated LETR with the newest meta-Acon activation function. We tested our model on the self-made dataset LDEG2021 on a single GPU and got an mAP of 91.9%, which surpassed our baseline model, Deformable DETR, by 1.1%. Furthermore, we modified LETR's convolution layer and named the new model after Ghost Leading/trailing Edge Transformer (GLETR) as a lightweight model for real-time detection. It is proved that GLETR has fewer weight parameters and converges faster than LETR with an acceptable decrease in mAP (0.1%) by test results. The proposed models provide the basis for subsequent parameter extraction work in the reconstruction area.

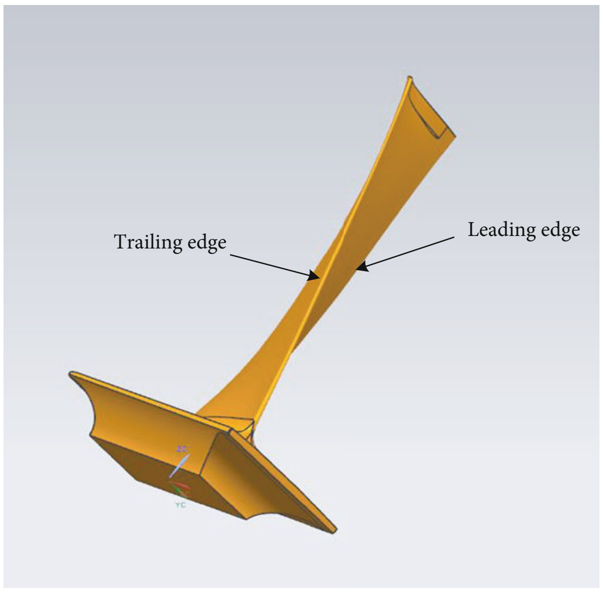
1. Introduction

The near-net-shaped blades are applied to the blades of the aero-engine as it fits the modern aero-engine performance better. The blank material and typical structure of the near-net-shaped blade are shown in Figures 1(a) and 1(b), respectively. A section curve of the blade is presented in Figure 2. The geometric parameters of the near-net-shaped blade's suction/pressure surface have met the designing requirement after being forged, which means that it needs no further machining, while the leading/trailing edge cannot be forged precisely due to the sharply changing curvature. On the other hand, although blank material is forged within the design tolerances, complex deformation still occurs [1]. That means we cannot plan the tool path according to the designed model. Hence, we need to reconstruct the theoretical leading/trailing edge. In this case, adaptive machining [2] is imported to the machining process of near-net-shaped blades. Adaptive machining technology aims to modify manufacturing data on the basis of changed conditions. In our previous manual work, we retained a part of the theoretical leading/trailing edge and bridged it with blank material considering the design intent and aerodynamic performance. The reconstructed blade's section curves are shown in Figure 3. This process, however, is time-consuming and depends on the human experience. Deep learning has a cutting-edge advantage in improving efficiency and avoiding human error, based on which we proposed a model reconstruction framework in [3].

Unlike traditional reconstruction methods based on geometric prediction, we reconstructed models based on the accomplished reconstruction stored in images. Our method



(a) The blank to be formed



(b) The near-net-shaped blade

Figure 1: The typical structure of near-net-shaped blade.

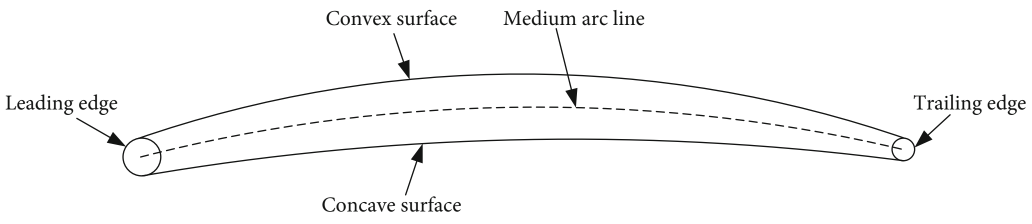


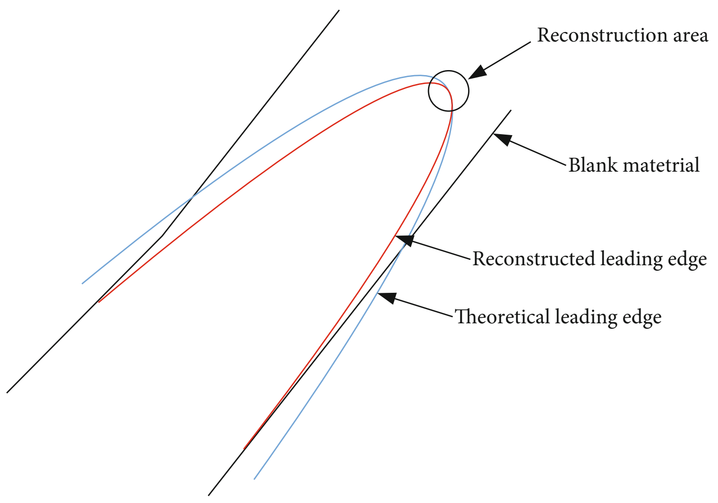
Figure 2: A section line of the near-net-shaped blade.

can be described in the following steps. In the first stage, we use Generative Adversarial Networks (GAN) [4] to classify optimal reconstructed leading/trailing edge's images based on our previous manual works [3]. However, the reconstructed curves in images cannot be adopted in computer-aided design (CAD) software for model reconstruction directly. So, we need to detect the retained part and bridged curves of these reconstructed curves on the next step. Thirdly, we will extract parameter information from them. As long as these parameters are obtained, we can utilize CAD software to adjust the theoretical leading/trailing edge and reconstruct it automatically to realize the adaptive machining of the near-net-shaped blade. This paper's main focus is to detect the retained part of the theoretical leading/trailing edge, and we use the item "reconstruction area" to represent it.

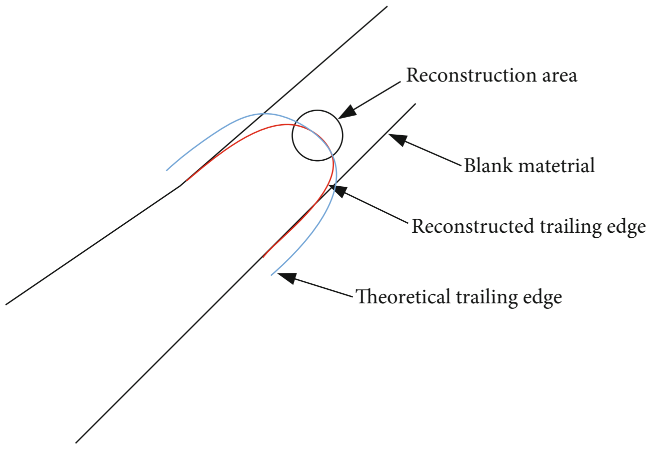
A small object usually contains a small quantity of semantic information due to its small size. As shown in Figure 3, in comparison with the general object detection task, there is no complex feature in the leading/trailing edge's image. Moreover, the leading/trailing edge's image contains less semantic information, and the background is relatively simple. For this reason, the leading/trailing edge's reconstruction area detection is defined as a small object detection task in this paper, and the reconstruction area is approximated by the area of the bounding box.

The performance of object detection has improved significantly with the help of convolutional neural networks (CNNs) [5], which detect objects by extracting features from considerable data. Generally speaking, object detection algorithms employing CNNs are divided into two-stage methods and one-stage methods according to their processing stages. A typical method of two-stage is the Regions with CNN features (R-CNN) series [6–8], which imports the selective search algorithm to predict the region of interest. Unlike two-stage methods, one-stage methods predict bounding boxes and classes in a single neural network. Symptomatic one-stage methods include You Only Look Once (YOLO) series [9–12] and Single-Shot Multibox Detector (SSD) [13].

The methods mentioned above have three points to be further ameliorated. First, the anchor box size needs to be manually designed for different detection tasks, which takes a great amount of time. Furthermore, the sizes of ground-truth bounding boxes of small objects are relatively small, which may lead to the class imbalance problem [14]. Moreover, the nonmaximum suppression (NMS) algorithm is not sensitive to small objects which contain less semantic



(a) Leading edge's reconstruction area



(b) Trailing edge's reconstruction area

Figure 3: The reconstruction area of a blade.

information. The number of the network's predictions is much larger than that of the actual objects' number, and NMS is employed here to remove superfluous predicted bounding boxes. The NMS algorithm sets a threshold value and deletes all bounding boxes higher than the threshold value and keeps the bounding box with the highest score. The postprocess of NMS during the small object detection is often obstructed by background, and small targets are easily covered by objects of medium and large sizes.

Considering the variable shape of the leading/trailing edge, we decided to take an anchor-free model as our leading/trailing edge detector, which abandons anchors and the postprocessing of the NMS algorithm. In recent years, Transformer has shown its superiority in the object detection field due to its capacity for spatial relation modeling of targets. The model that works best now with Transformer is Deformable DETR (Deformable Detection Transformer) [15]. Nevertheless, Deformable DETR has its issues. Firstly, Deformable DETR extracts features from input images based on the spatial attention mechanism. That means some valuable information is abandoned during this process. Its accuracy, therefore, remains a huge space to be promoted. Then, despite that Deformable DETR has achieved prominent results, it still needs to be further compressed to meet the requirement of real-time detection. Focusing on these two abovementioned problems, the main works of this study are summarized below:

(1) We proposed an anchor-free model named Leading/trailing Edge Detection Transformer (LETR). LETR extracts features of reconstruction areas from the frequency-channel mixed domain. Besides, LETR activates nonlinear units dynamically in view of the simplex background of our dataset

(2) To balance model weight and performance, we introduced a lightweight model by modifying LETR for real-time detection. Technically, the specific convolutional layers of LETR are replaced by lightweight modules. The model is called Ghost Leading/trailing Edge Detection Transformer (GLETR).

(3) We tested LETR and GLETR on the self-made dataset LDEG2021. It is proved that LETR has state-of-the-art performance on detecting reconstruction

areas. Moreover, GELTR has significantly fewer parameters and converges faster than LETR in the training process

The rest of this paper is structured as follows. Section 2 illustrates the recent works on the reconstruction of edge shape in adaptive machining and object detection. Section 3 reviews the attention mechanism in computer vision. Our proposed models are presented in Section 4. Section 5 reports the experimental results on the self-made dataset. Section 6 gives the conclusions and future works. The code will be available at https://github.com/andrewsilver1997/LETR.

2. Related Works

Our research is the combination of adaptive machining and computer vision algorithms. In this section, we will discuss the related works from the aspects of edge shape reconstruction, small object detection, self-attention mechanism, activation function, and efficient networks.

2.1. Reconstruction of Edge Shape. The prevalent model reconstruction methods can be concluded in three aspects: by fitting curves of measured data, predicting the deformation of the blade's surface, and predicting design intent. Our reconstruction work is mainly based on design intent and deformation of blank material. A model similar to the original counterpart is constructed according to the existing data in the reconstruction of the edge shape procedure. Some representative research is illustrated as follows. Yun et al. [16] imported parameterization design to Free-Form Deformation (FFD) and realized the model reconstruction based on measure data. Zhao et al. [17] improved the accuracy of reconstruction by inserting knots. Feng et al. [2] predicted the desired spline curves considering the deformation and thickness of the blank material while preserving the design intent. Yu et al. [18] established the relationship between measured points and the velocity field of the blade's section. Further, they calculated the curves in the area without measuring data. Zhang et al. [19] considered the constraints of the chord length, angle of attack, and radius during the reconstruction process. Wu et al. [20] proposed a novel reconstruction algorithm by removing bad measure points and optimizing the iterative closest point (ICP) algorithm. Nevertheless, the current research on the modeling of large curvature inexact forming regions mainly focuses on simple geometric element fitting but does not fully consider the relationship between the design intention of the blade, the similarity relationship between the actual blade surface and theoretical surface, and the complex deformation of blank materials. Our previous manual works addressed the aforementioned issue to a certain extent [3]. For this article, we further proposed an algorithm based on deep learning to accelerate this process and reduce errors caused by human experiences.

2.2. Small Object Detection. It is usually a small part of the theoretical model and has less semantic information in an image. Hence, the detection of the reconstruction area can be seen as a small object detection task. Some models based on the two-stage method were proposed to detect small objects. Based on Faster R-CNN [7], Singh and Davis [21] proposed Scale Normalization for Image Pyramids (SNIP) for small object detection. Furtherly, they modified SNIP and proposed SNIPER [22] for efficient and fast detection. Zhang et al. [23] utilized a multiscale feature fusion layer (MFL), and a certain extent of improvement was obtained. On the other hand, a typical one-stage detector is RetinaNet [14], which uses focal loss and feature pyramid maps [24] to detect small objects. Based on YOLO v2, the work of Liu et al. [25] can detect arbitrary-oriented targets of small sizes.

Some researchers proposed anchor-free methods. The representative works include CornerNet [26], ExtremeNet [27], and CenterNets [28, 29]. These models detect objects by implementing keypoint estimation. Some novel works aiming to solve the problems of the traditional NMS algorithms were published. Dong et al. [30] integrated transfer learning with faster RCNN for annotation and improved accuracy. Cai and Vasconcelos [31] proposed cascade RCNN. Cascade RCNN connects a sequence of detectors and adapts threshold in NMS processing to avoid the mismatch between predicted bounding boxes and ground-truth objects. Our models discard anchor boxes and NMS postprocessing. Experiments verified that our models dominate in existed models without the help of the anchor box and NMS algorithm.

2.3. The Self-Attention Mechanism in Computer Vision. In recent years, the progress of natural language processing [32] (NLP) has led researchers to investigate its application in computer vision. Vaswani et al. [33] introduced Transformer in NLP for the first time. The Transformer achieves impressive results on sequence prediction. Carion et al. [34] modified the Transformer and named their model as Detection Transformer (DETR). The DETR model abandons anchor boxes and NMS's implementation. However, DETR has a low speed of convergence and does not perform well on small object detection tasks. Sun et al. [35] argued the cross-attention module of DETR is not necessary and proposed two models named Transformer-based Set Prediction with FCOS [36] (TSP-FCOS) and Transformer-based Set Prediction with R-CNN (TSP-RCNN), respectively. Zhu et al. [15] imported deformable convolution [37] to DETR and proposed Deformable DETR. Deformable DETR deals with input feature maps from the spatial domain. In this study, we tried to extract features from a different aspect, that is, the mixed domain of frequency and channel.

2.4. Activation Functions. Some researchers attempted to modify the activation function in networks and gained improvements to a certain extent. The most-used activation function is Rectified Linear Unit (ReLU) [38] which activates the neurons linearly when the input is bigger than zero. Its variant, Leaky ReLU [39], activates the neurons when the input is nonzero. The ReLU-based activation functions are too simple to be implemented on complicated visual tasks as some features may be missed. Sigmoid-weighted Linear Unit (SiLU) [40] combines ReLU and sigmoid function together. Compared with ReLU, it is smooth and nonmonotonic. [41] applied automatic search to a variety of activation functions and found a better function named Swish. But it also brings huge computation resource consumption. [42] imported spatial context to the activation function and gained state-of-the-art performance on dense visual tasks. A recent work is called Dynamic Rectified Linear Unit (DY-ReLU) [43] adapts the parameters dynamically by a hyperfunction. When implemented on deep neuron networks, however, the abovementioned activation functions' effects become weaker. Unlike previous activation functions, Activate or Not functions (Acon) [44] learn whether to activate the specific neurons and convert models into dynamic networks, which brings better performance as networks go deeper.

2.5. Efficient Networks. For deploying object detection models on mobile devices, some operations for light-weighting are inevitable to save computational cost and memory. There are two types of strategies for efficient networks. One is to compress the model by pruning redundant connections [45] and channels [46], quantization [47], and knowledge distillation [48]. Such model compression methods usually require well-designed architecture and pre-trained models. The other strategy is to design a lightweight model directly [49]. Some typical lightweight models are proposed by researchers. Xception [50] introduced depth-wise separable convolution modules and realized better performance with fewer model parameters. MobileNet series [51–53], for example, are based on depth-wise and point-wise separable convolutions as well as automated machine learning (AutoML) technologies [54]. Other famous light-weighted models are ShuffleNet [55, 56] series, which exchange their inner channel information to reduce computational cost. The redundancy of feature maps has not been solved well. GhostNet [49], on the contrary, utilizes these redundant feature maps and has more excellent performance compared with previous lightweight models. We integrated our LETR with GhostNet and presented GLETR. More details of GLETR are seen in Section 4.

3. Revisiting Deformable Attention

3.1. Self-Attention Mechanism. DETR is a successful object detection model using the self-attention mechanism, achieving outstanding performance on the COCO dataset [57]. In this part, we briefly reviewed the inner mechanism of DETR.

3.1.1. Multihead Attention. In the natural language processing field, the self-attention mechanism was adopted in the Transformer model. By computing the values of Query, Key and Value of input images, the Transformer modulates the compatibility of every pixel. In the multihead attention mechanism, the outputs of several attention heads are aggregated linearly with learnable weight parameters. Given the Value's input feature z_q and the Query and Key's input feature $x \in HW \times C$, where H, W, and C are the heights, widths, and channels of input images, the formulation of multihead attention is shown as:

$$\mathrm{MultiHeadAttn}\left(z_q, x\right) = \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_k} A_{mqk} \cdot W'_m x_k\right]. \quad (1)$$

In Equation (1), $q \in \Omega_q$ and $k \in \Omega_k$ are elements of Query and Key set, m is the number of attention heads, $A_{mqk} \propto \exp\{z_q^T U_m^T V_m^T x_k / \sqrt{C_v}\}$ is attention weight, where U_m and V_m are the transformation matrices of Query, Key, respectively, and $C_v = C/M$. W_m and W'_m are transformation matrices of Value. Please note that U_m, V_m, W_m, and W'_m's parameters are all learnable.

3.1.2. Position Embedding. DETR adopts position embedding to our extracted features. This is owing to that the Transformer demands information about the relative or absolute position of pixels in feature maps. In DETR, sine and cosine functions are used to represent the positions of different pixels. The position embedding equations are written as:

$$\begin{aligned} PE(\mathrm{pos}, 2i) &= \sin\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right), \\ PE(\mathrm{pos}, 2i+1) &= \cos\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right), \end{aligned} \quad (2)$$

where pos denotes the position and i is the dimension index; d_{model} represents the channel dimension of the input features.

3.1.3. Match Loss. Unlike previous object detection models using NMS, DETR outputs fixed N predictions and N is much larger than the number of objects in an image. DETR needs to allocate predictions to objects in images, and match loss is introduced to evaluate this process. Assuming that y_i is the set of ground-truth bounding boxes, $y_{\sigma(i)}$ is a set of predicted bounding boxes with index $\sigma_{(i)}$, the optimal permutation σ is written as:

$$\sigma = \arg\min_{\sigma \in P_N} \sum_{i}^{N} L_{\mathrm{match}}\left(y_i, y_{\sigma(i)}\right), \quad (3)$$

where σ stands for a permutation of predicted objects. Finally, the match loss L_{match} is defined as:

$$L_{\mathrm{match}} = \sum_{i=1}^{N} \left[-\log P_{\sigma(i)} + L_{\mathrm{box}}\left(b_i, \hat{b}_i\right)\right]. \quad (4)$$

Albeit the fact that DETR has achieved state-of-the-art performance, it costs massive computational resources, and its convergence speed is relatively low. Furthermore, the performance of DETR on small target datasets is

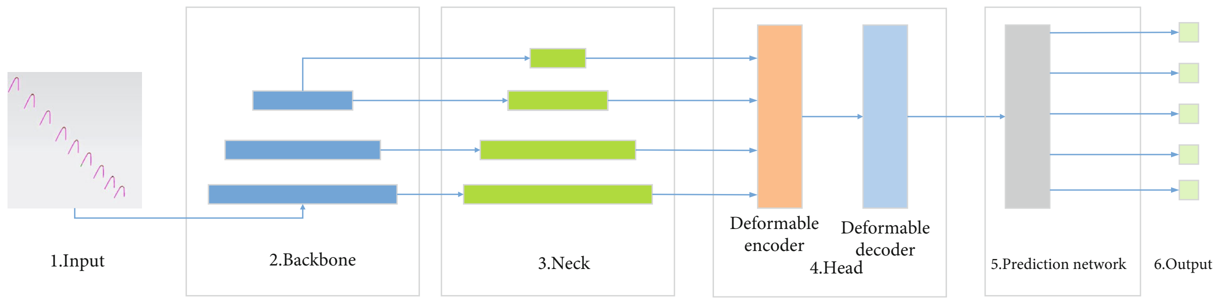


FIGURE 4: Overview of LETR.

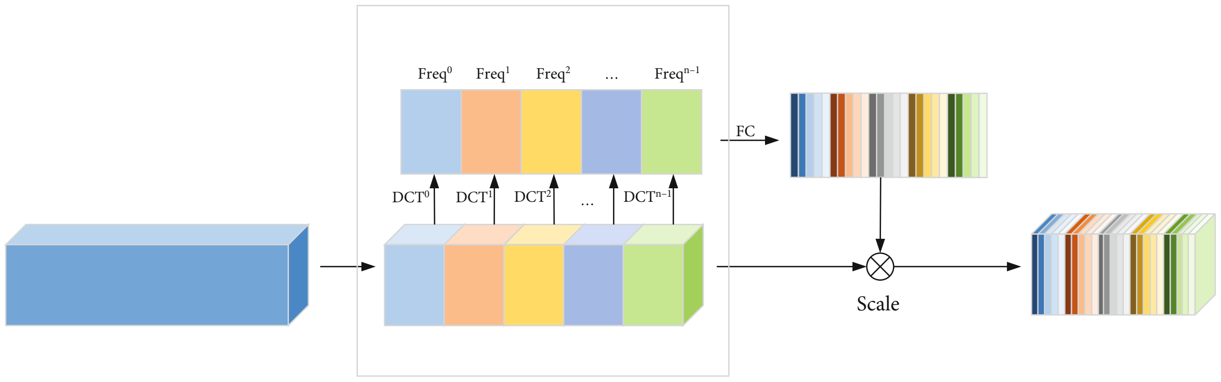


FIGURE 5: The feature extraction process of the multispectral channel attention module.

disappointing. Besides, the test results of DETR present no outstanding performance on our detection task. Hence, it is proved that DETR is not suitable for our detection task.

3.2. Deformable Attention. DETR did not perform well on small target detection tasks, owing to its inner mechanism. Zhu et al. [15] combined DETR with deformable convolution and named their model Deformable DETR, by which self-attention is generated to image deformable attention in the image domain. The feature of the deformable attention is computed by:

$$\text{DeformAttn}\left(z_q, p_q, x\right) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} \cdot W'_m x\left(p_q + \Delta p_{mqk}\right)\right]. \tag{5}$$

k indicates the sampled keys, and K means the number of sampled keys. p_q denotes the reference point, and Δp_{mqk} is sampling offset.

In small object detection tasks, the sizes of objects vary from small size to large size. To adapt the multiscale size of input features, the deformable attention generates to multiscale deformable attention and can be calculated by:

$$\text{MSDeformAttn}\left(z_q, \widehat{p}_q, x^l\Big|_{l=1}^{L}\right) = \sum_{m=1}^{M} W_m \left[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlpk} \cdot W'_m x^l\left(\left(\phi_q\right) + \Delta p_{mlqk}\right)\right], \tag{6}$$

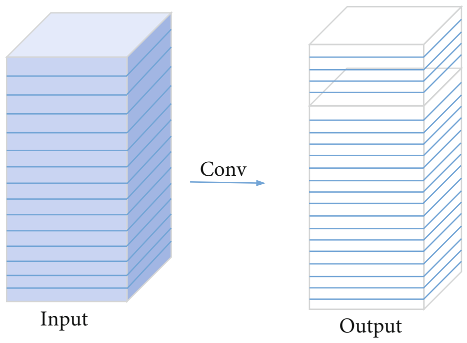
where $\widehat{p}_q$ is the coordinates of sampled points being normalized, l indexes the level of input feature maps. A_{mlqk} and Δp_{mlqk} represent the attention weight and offset of sampling for the l^{th} input feature map, respectively. $\phi_l(p_q)$ function rescales p_q according to the l^{th} input feature map.

Deformable DETR speeds up its convergence and makes progress on small targets datasets. For our leading/trailing edge's reconstruction area detection task, however, Deformable DETR's performance can be further improved compared with previous detectors using anchor box and NMS.

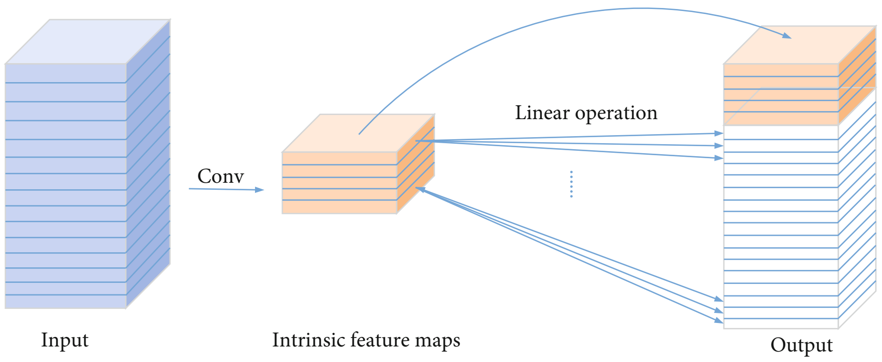
4. LETR and GLETR

4.1. Overview of LETR. As shown in Figure 4, our model adopted state-of-the-art methods implemented in small object detection. A frequency-channel mixed backbone (we named it after FcaAconNet) extracts features using a pyramid neck. And then the pyramid-shaped feature maps are sent to the Deformable Transformer head for encoding and decoding. A feed-forward network is configured to output the predicted classes and locations. We adapted the configurations of the deformable encoder, deformable decoder, and prediction network in [15].

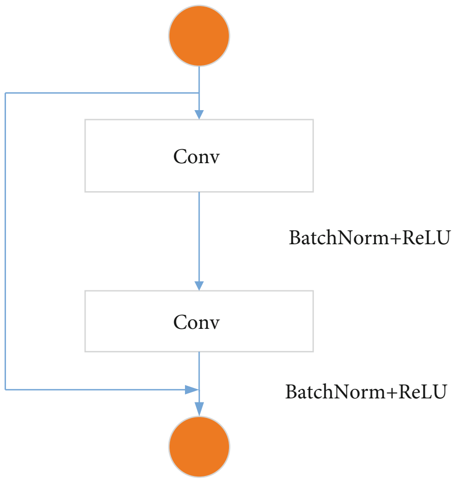
4.2. Feature Extraction. Technically, the backbone is responsible for extracting feature maps over input images. Previous



(a) The convolution operation



(b) The processing of ghost module



(c) The inner connection of the ghost module

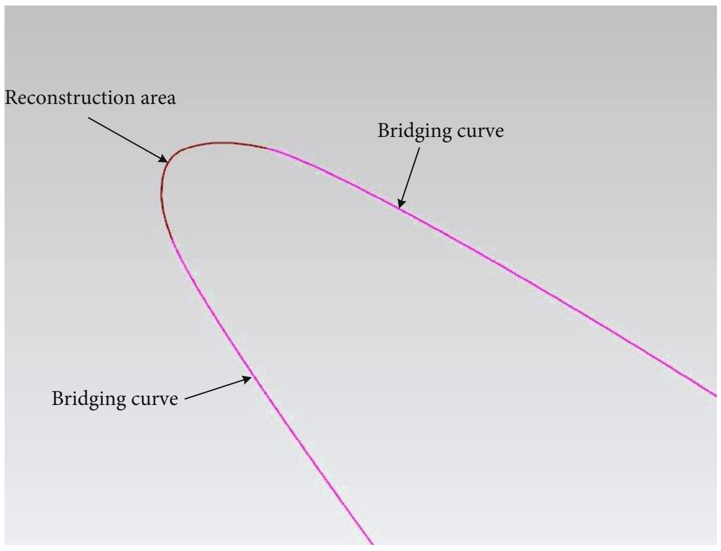
Figure 6: The ghost module in Ghost-FcaAconNet.

neural network models output a single-scale feature map. However, the reconstruction areas in our dataset vary in different sizes. Hence, we adopted the pyramid architecture in Deformable DETR for multiscale object detection. Furthermore, inspired by FcaNet [58], we imported the multispectral channel attention module of FcaNet to the feature extraction process of the model's backbone. The details of the utilized module are demonstrated in Figure 5.

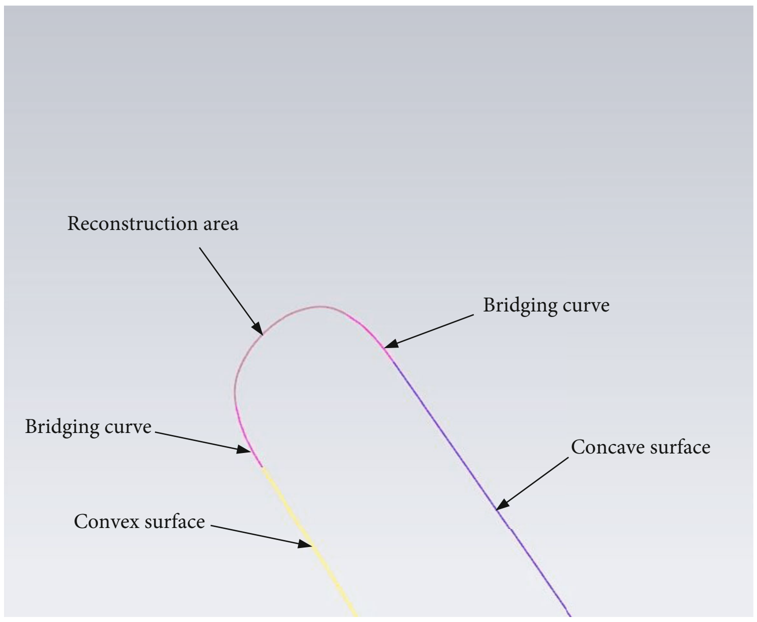
Assuming that the dimension of the input feature is $C \times H \times W$, the multispectral channel attention module transforms input features into frequency domain with the help of 2-dimensional discrete cosine transformation (2D DCT). 2D DCT is formulated by Equation (7):

$$\text{2DDCT}\left(x^{2d}\right) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} x_{i,j}^{2d} \cos\left(\frac{\pi h}{H}\left(i+\frac{1}{2}\right)\right) \cos\left(\frac{\pi w}{W}\left(j+\frac{1}{2}\right)\right), \tag{7}$$

where x^{2d} is the input image, and $h \in [0, 1, \cdots, H-1]$, $w \in [0, 1, \cdots, W-1]$.



(a) Leading edge



(b) Trailing edge

Figure 7: Selected samples in LDEG2021.

In FcaAconNet, the feature extraction procedure is illustrated as follows. First, input features X are split into n parts according to the channel. This process can be denoted as:

$$X = \left[X^0, X^1, \cdots, X^{n-1}\right]. \tag{8}$$

Next, the ith frequency feature of a given input X^i is computed by the following formulation. $2DDCT$ is represented for 2D DCT operation.

$$\mathrm{Freq}^i = \mathrm{2DDCT}\left(X^i\right). \tag{9}$$

Thus, by concatenating all Freq^i, we have the final frequency feature Freq of the input feature:

$$\mathrm{Freq} = \mathrm{cat}\left(\left[\mathrm{Freq}^0, \mathrm{Freq}^1, \cdots, \mathrm{Freq}^{n-1}\right]\right). \tag{10}$$

And finally, the attention of the mixed frequency-channel domain is computed by the following formulation:

$$\mathrm{MSAttn} = \mathrm{sigmoid}(\mathrm{fc}(\mathrm{Freq})), \tag{11}$$

where sigmoid is sigmoid function and $fc(\cdot)$ is a fully connected layer. By rescaling X with $MSAttn$, the output of

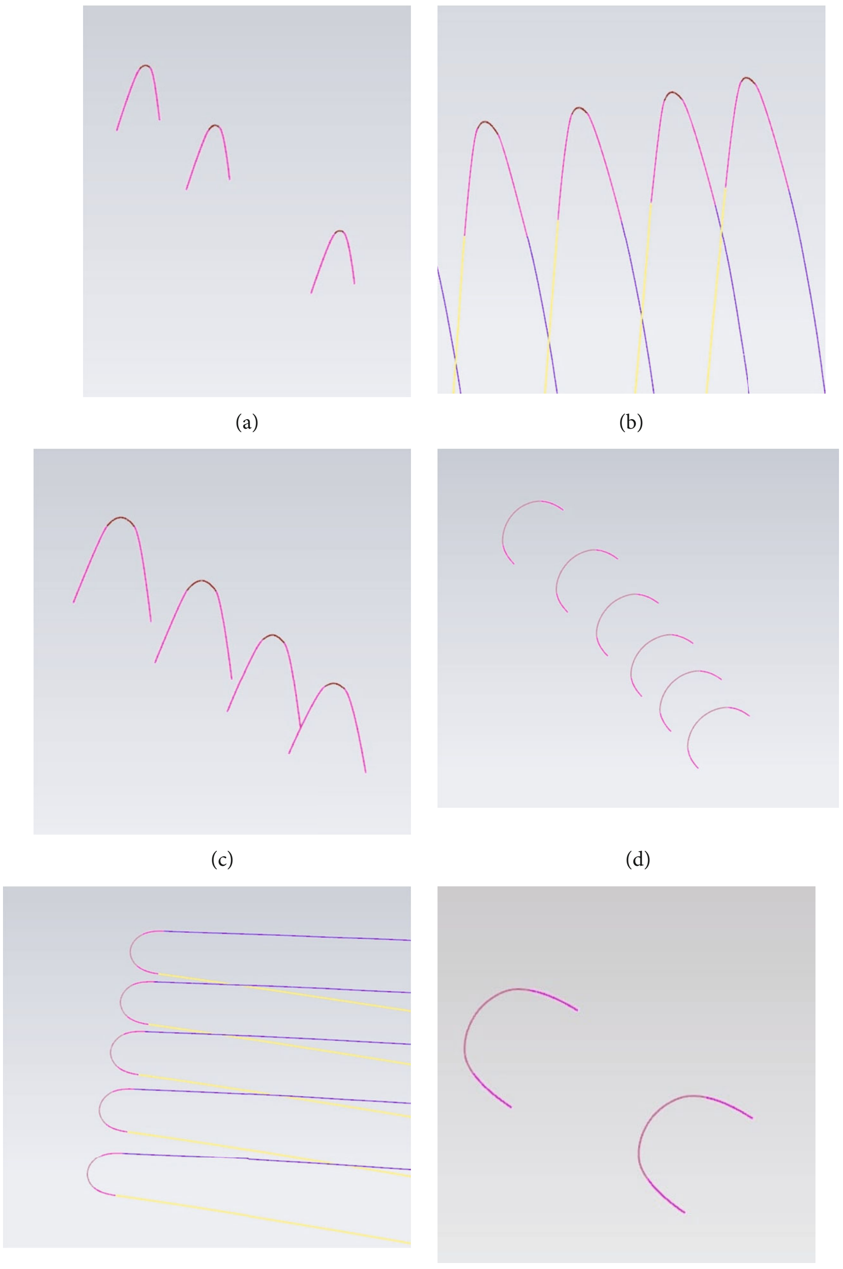

(a) (b) (c) (d) (e) (f)

Figure 8: Examples of LDEG2021 dataset.

multispectral channel attention module, denoted as $\tilde{X}$, is computed as:

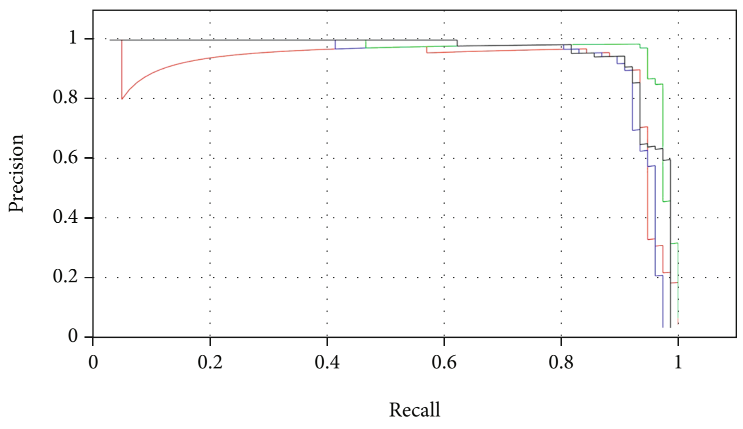
$$\tilde{X} = F_{\text{scale}}(X, \text{MSAttn}) = \text{MSAttn} \times X. \quad (12)$$

The mixed frequency and channel weights are applied to the input feature maps after the multispectral channel attention module. In this method, adequate frequency and channel information is utilized by our backbone, and the accuracy is enhanced gradually in deep neural networks.

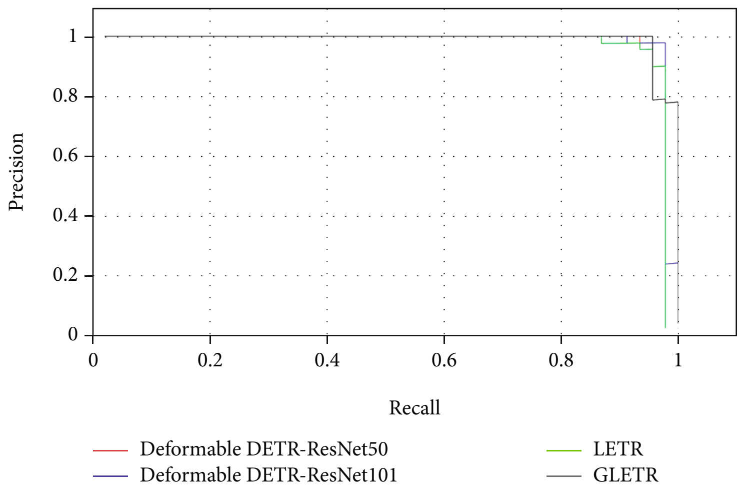
4.3. Activation Function. In the neural network's architecture, the activation function plays a key role in importing nonlinearity to improve the model's classification capability. Considering the big difference between the foreground and background of our task, we employed Acon functions because of their excellent ability to control the extent of

Table 1: Confusion matrix.

Ground-truth	Prediction	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN



(a) Comparisons of PRCs on leading edges



(b) Comparisons of PRCs on trailing edges

Figure 9: Comparison of PRCs.

nonlinear activation. In other words, by learning whether to activate and to what extent the input is activated, the model filters some disturbing information. More concretely, Acon functions are divided into three types: Acon-A, Acon-B, and Acon-C. Acon-A and Acon-B can be seen as special cases of Acon-C. According to Ma et al. [44], meta-Acon, the variant of Acon-C, showed the best performance in the test. Therefore, we adopted it as our activation function. Here, we gave the definition of Acon-C and meta-Acon.

Firstly, we use a function s_β to approximate the general maximum function max $(x_1, x_2, \cdots, x_n)$:

$$s_\beta(x_1, x_2, \cdots, x_n) = \frac{\sum_{i=1}^{n} x_i e^{\beta x_i}}{\sum_{i=1}^{n} e^{\beta x_i}}, \tag{13}$$

where β is the switching factor. Next, we consider the situation where $s_\beta(x_1, x_2, \cdots, x_n)$ is in neural networks. In this case, $s_\beta(x_1, x_2, \cdots, x_n)$ degenerizes to the following format:

$$s_\beta(\eta_a(x), \eta_b(x)) = (\eta_a(x) - \eta_b(x)) \cdot \sigma[\beta\eta_a(x) - \eta_b(x)] + \eta_b(x). \tag{14}$$

σ is the sigmoid function, $\eta_a(x)$ and $\eta_b(x)$ represent linear functions. Considering a more general situation where $\eta_a(x) = p_1 x$ and $\eta_b(x) = p_2 x$, Acon-C is written as:

$$\mathrm{Acon} - C(x) = s_\beta(p_1 x, p_2 x) = (p_1 - p_2)x \cdot \sigma[\beta(p_1 - p_2)x] + p_2 x. \tag{15}$$

And furtherly, we see β as a learnable network $G(x)$ and $Acon - C(x)$ can be generated to $\mathrm{meta} - \mathrm{Acon}(x)$, which is computed as:

$$\mathrm{meta} - \mathrm{Acon}(x) = (p_1 - p_2)x \cdot \sigma[G(x)(p_1 - p_2)] + p_2 x. \tag{16}$$

We replaced the first two ReLU functions with meta-Acon in each bottleneck of the original FcaNet to avoid overfitting and proposed FcaAconNet. Relative experiments are seen in Section 4.

As Figure 6(a) shows, convolution operation generates a great amount of output feature maps which contains a certain extent of redundancy. Han et al. proved that some similar feature maps exist in this redundancy and argued that these superfluous feature maps are the ghost of intrinsic

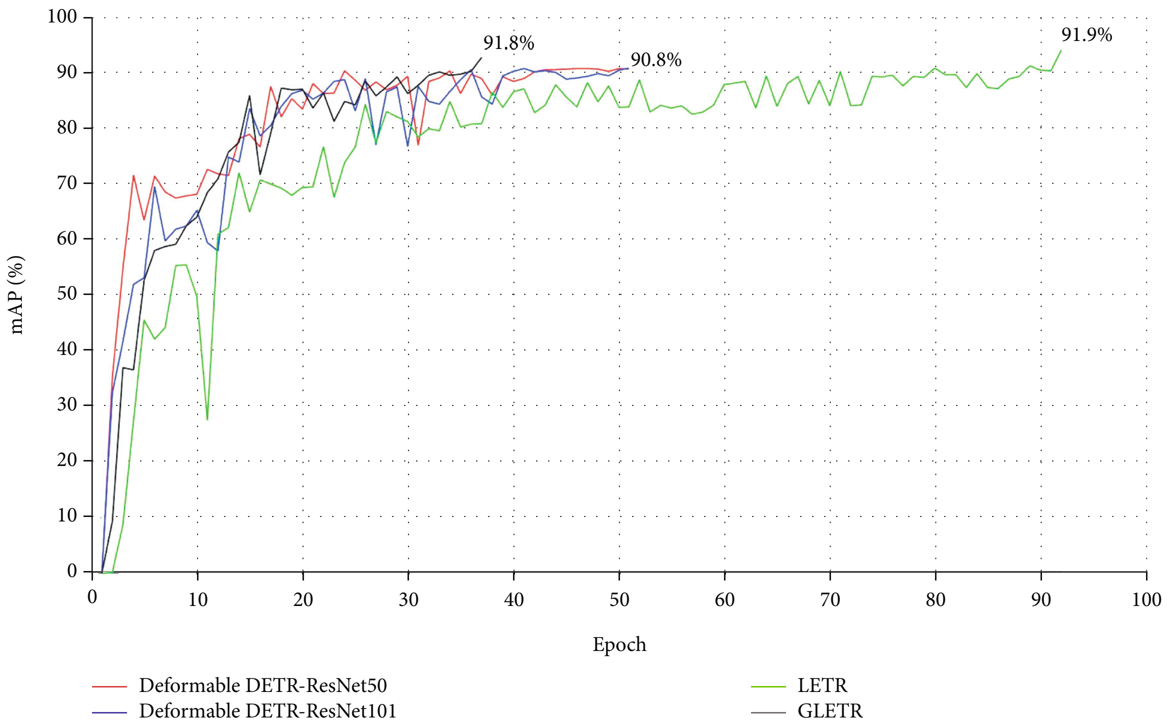


FIGURE 10: Convergence curves of three models on LDEG2021.

TABLE 2: Comparison of LETR and GLETR with Deformable DETR on the LDEG2021 test set.

Detector	Backbone	mAP	Model weight	Training hours
Deformable DETR	ResNet-50	90.6%	491.2 MB	3.5
	ResNet-101	90.8%	718.9 MB	3.5
LETR	FcaAconNet	91.9%	545.6 MB	8.5
GLETR	Ghost-FcaAconNet	91.8%	424.4 MB	3.5

feature maps. Thus, they generated intrinsic feature maps by a primary convolution and obtains ghost features through a linear operation, by which the complexity of the model is reduced. The whole procedure is described as follows.

4.4. Ghost Module. If we use X to represent the input feature map, the output Y after general convolution is defined by:

$$Y = X * f + b, \tag{17}$$

where $*$ and b denote the convolution operation and bias, respectively, and f is the convolution operator. In ghost module's first stage, however, f is replaced by a new operator f' and bias is canceled for lower model complexity. Thus, the output Y' of the ghost module's first stage is denoted as:

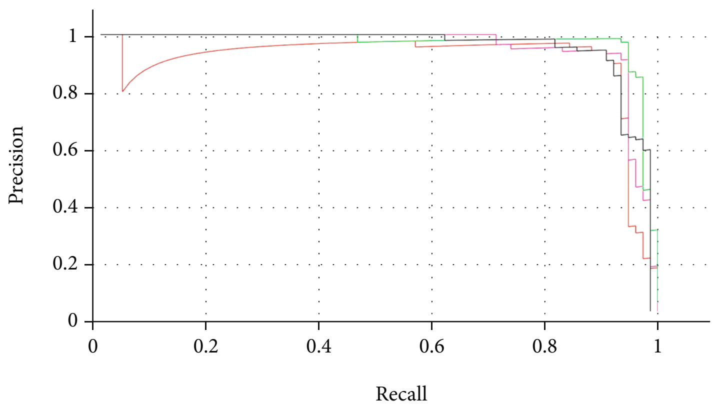
$$Y' = X * f'. \tag{18}$$

In the second stage, the ghost module implements a series of linear operations to output Y' to match the dimension of the channel of the original output Y. The linear operations in the ghost module are written as:

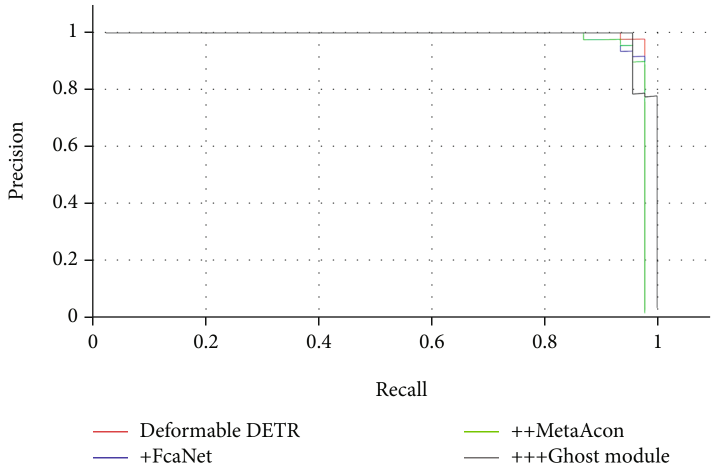
$$y_{i,j} = \phi_{i,j}\left(y_i'\right), \tag{19}$$

where y_i' is the i^{th} intrinsic feature map, $\Phi_{i,j}$ represents the i^{th} linear operation, and $y_{i,j}$ is the generated ghost feature map.

The details of the ghost module are depicted in Figure 6(b). Some intrinsic feature maps are output by the previous convolution layer in the first stage. Next, by linear operation, we mentioned above, a large number of ghost feature maps are produced. Finally, the intrinsic feature maps and their corresponding "ghost" are concatenated according to the channel dimension. In practice, the inner connection of the ghost module is shown in Figure 6(c). The linear operation is carried out by a convolution layer. Technically, we replaced all convolution layers with ghost modules in FcaAconNet's bottleneck, and we called the FcaAconNet backbone with ghost module Ghost-FcaAconNet.



(a) PRCs of the models with different components on the leading edges



(b) PRCs of the models with different components on the trailing edges

FIGURE 11: PRCs of the models with different components.

5. Experiments

5.1. Dataset. We made LDEG2021 to detect the reconstruction area of the leading and trailing edge. The whole images of LDEG2021 are the multiple sizes of section curves of different reconstruction areas of the leading and trailing edges of the different blades' different heights. These images are acquired from the screenshots of Unigraphics NX 7.5 (UG) software. LDEG2021 has 397 images annotated with two classes: leading edge and trailing edge. Two selected samples and their corresponding captions in LDEG2021 are shown in Figure 7. We used different colors to distinguish different curves. The magenta curves in Figures 7(a) and 7(b) are the bridge curves of reconstructed leading/trailing edges. And the crimson curves indicate the reconstruction areas of leading edges. On the other hand, the plum curves are the reconstruction areas of trailing edges. The section lines of convex and concave faces are presented in purple and yellow.

The reconstruction areas vary in different sizes. So, we applied scaling and rotation to the models of reconstruction areas. In this way, some screenshots of reconstruction areas are no longer in actual sizes. We need to notice that this process is to train the ability to detect shape-varied reconstruction areas. In Figure 8, we also gave other examples of the LDEG2021 dataset. Figures 8(a)–8(c) present the images of reconstructed leading edges. Figures 8(d)–8(f) are the images of reconstructed trailing edges. In this article, we aim to train a high-performance model which can detect a certain number of reconstruction areas at the same time. That explains why there are more than one leading/trailing edge in an image. It is also worth noting that some overlapping phenomena occurred in Figures 8(b) and 8(e). This is due to the fact that these images are obtained from different perspectives in UG. The LDEG2021 dataset will be open-sourced on https://github.com/andrewsilver1997/LDEG2021.

5.2. Data Augmentation. Generally, a neural network model gets better performance when the dataset's scale gets larger. The number of images in LDED2021, nevertheless, is limited. The data augmentation methods we applied include random flipping, random cropping, and resizing. The training procedure was carried out on a CPU of Intel Xeon E5-2678 V3 and a single GPU of Nvidia RTX 2080Ti. We used MMDetection object detection toolbox [59] with Pytorch 1.5.1, cuDNN 7.6.1, and CUDA 10.1 for implementation.

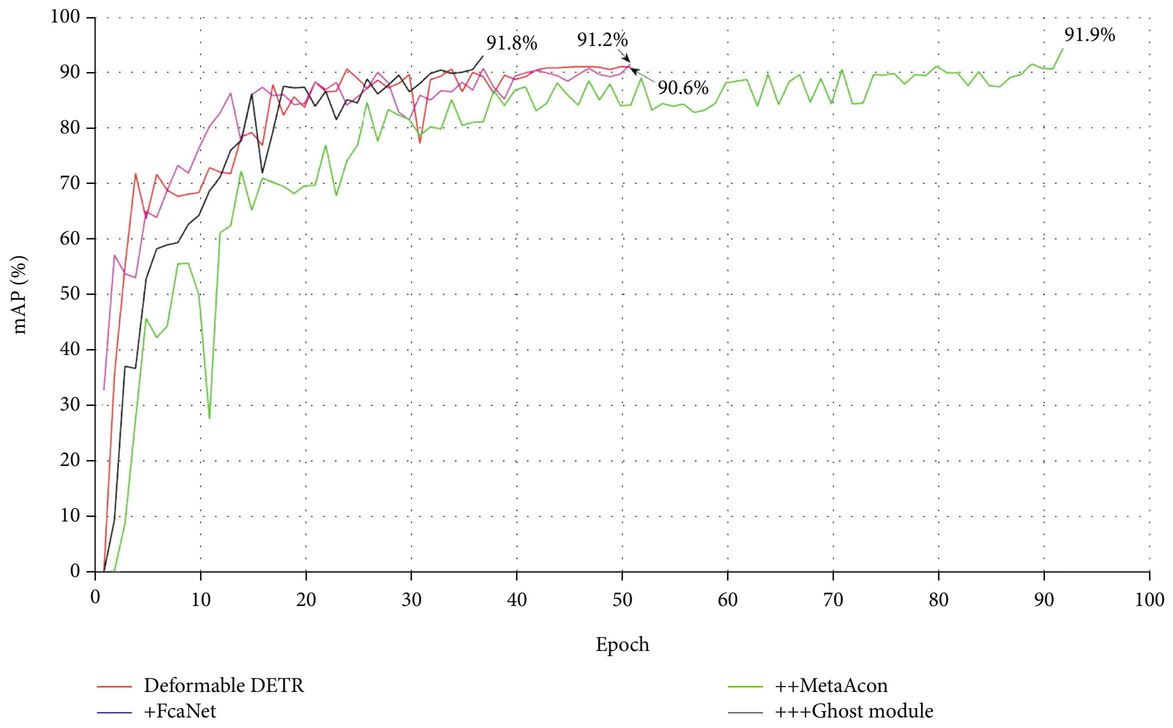


Figure 12: Convergence curves of the models with different components.

Table 3: Ablation studies for FcaNet, meta-Acon, and ghost module.

ResNet	FcaNet	Meta-Acon	Ghost module	mAP	Model weight	Training hour
✓				90.6%	491.2 MB	3.5
	✓			91.2%	543 MB	3.5
	✓	✓		91.9%	545.6 MB	8.5
	✓	✓	✓	91.8%	424.4 MB	3.5

The batch size and learning rate were set to 2 and 2×10^{-4} at first, respectively. The learning rate was decreased to 2×10^{-5} after 40 training epochs. Moreover, we chose adamW as our optimizer.

5.3. Results and Discussions

5.3.1. Evaluation Metrics. True positive (TP), false positive (FP), false negative (FN), and true negative (TN) are usually needed in the evaluation of a model's performance. Table 1 shows their definition by giving a confusion matrix.

We utilized the precision-recall curve (PRC) and mAP (mean average precision) to evaluate the proposed approach's performance. Firstly, we give the definition of precision and recall:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{20}$$

Generally speaking, a good object detection model is supposed to raise its precision and keep the recall at a relatively high level.

AP (average precision) represents the performance that the model detects a specific class of all objects. AP is defined in

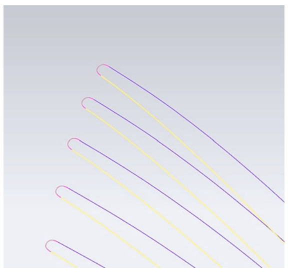
$$AP = \int_0^1 p(r)sdr, \tag{21}$$

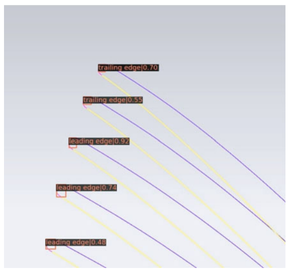
where p and r are short for precision and recall. mAP (mean average precision) is defined as the average of all classes' AP. It is calculated by

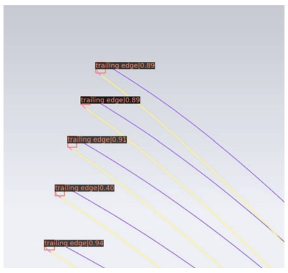
$$mAP = \frac{\sum_{i=1}^{N} \text{AP}_i}{N}, \tag{22}$$

where AP_i is the AP value of the i^{th} class and N indicates the total number of all classes. mAP measures the overall performance of the model. In other words, the higher the mAP is, the higher accuracy our models have. The other metrics we implemented are model weight and training hour, which measure the number of weight parameters and convergence speed.

5.3.2. Results on LDEG2021 Dataset. Figures 9(a) and 9(b) show the PRC curves of Deformable DETRs with ResNet50 and ResNet101 backbone, LETR, and GLETR. The performances of the Deformable DETRs with ResNet50 and ResNet101 were nearly the same when detecting the reconstruction areas of the leading edge. Maintaining the same


(a) Raw image


(b) Detection results of Deformable DETR


(c) Detection results of LETR

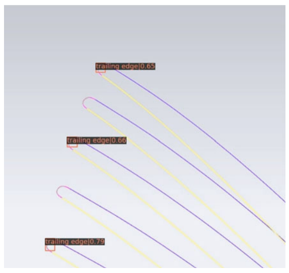
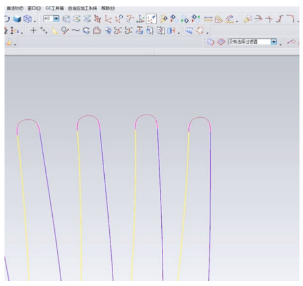
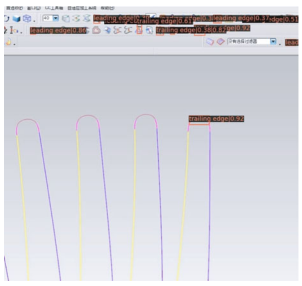
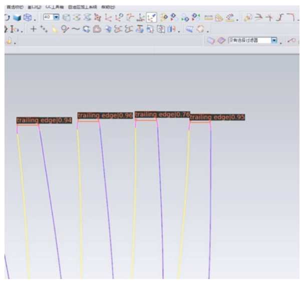

(d) Detection results of GLETR

Figure 13: Selected examples from the detection results on trailing edges.


(a) Raw image


(b) Detection results of Deformable DETR


(c) Detection results of LETR

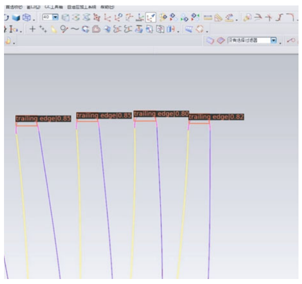
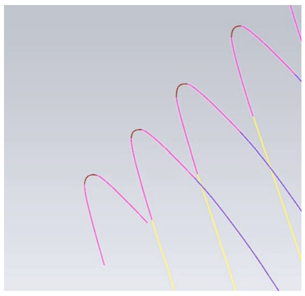
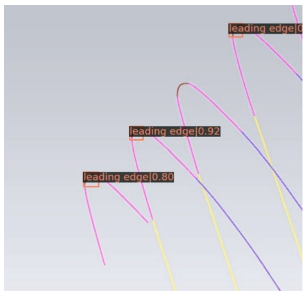

(d) Detection results of GLETR

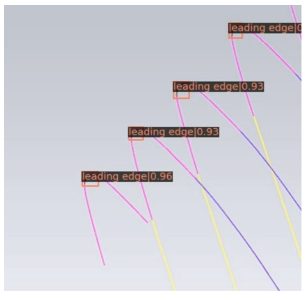
Figure 14: Another examples from the detection results on trailing edges.

recall, the precision values of LETR and GLETR were higher than Deformable DETRs. The enhancement of the performance when detecting the trailing edge' reconstruction area was unobvious as shown in Figure 9(b). It is worth noting that the shapes of trailing edge's reconstruction areas have no such big differences as leading edge's reconstruction areas do. Besides, the retained theoretical parts of the trailing edge's reconstruction areas are relatively larger than that of the leading edge's reconstruction areas. That explains why the improvement in Figure 9(b) is not obvious. But even though the improvement in Figure 9(b) is not distinct, LETR and GLETR still show their comparable performance. We will give a more detailed illustration in the following discussions of detection results. The convergence curves of the three models are presented in Figure 10. Deformable DETR with ResNet50 backbone took 50 epochs to converge and the mAP was 90.6%. The Deformable DETR with ResNet101 backbone did not show significant differences compared with the Deformable DETR with ResNet50. On the other hand, LETR, achieved a higher mAP even though its training time was longer. GLETR had the fastest convergence speed (less than 40 epochs) and the mAP was 91.8%.

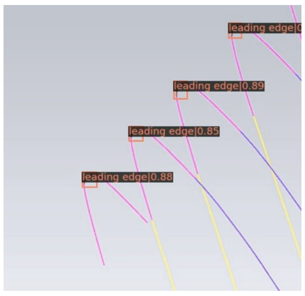

(a) Raw image



(b) Detection results of Deformable DETR

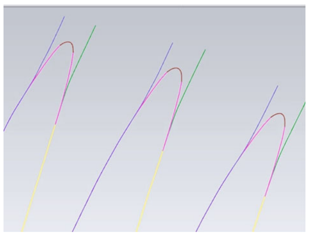


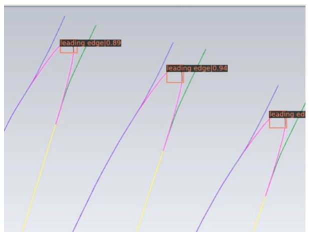
(c) Detection results of LETR

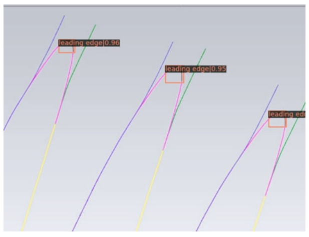


(d) Detection results of GLETR

Figure 15: Comparison of detection results of the top area.


(a) Raw image


(b) Detection results of Deformable DETR


(c) Detection results of LETR

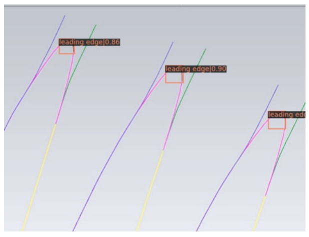
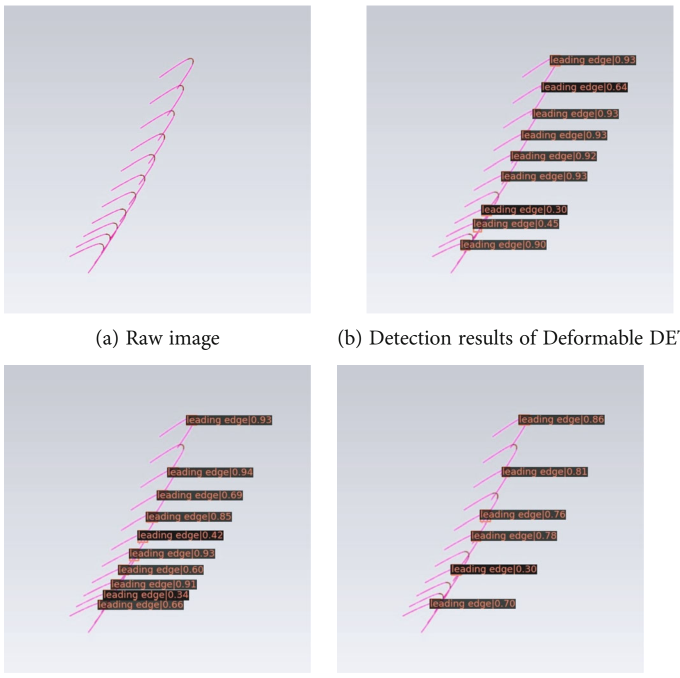

(d) Detection results of GLETR

Figure 16: Comparison of detection results of the middle area.

Table 2 illustrates the results of the test. Compared with Deformable DETR, LETR achieves better performance with a 1.1% improvement of mAP. On the other hand, GLETR decreased model weight by 121.2 MB with a slight decrease of mAP (0.1%). It is worth noting that the results were obtained on a single GPU. We believe the improvement in the performance of our models will be more pronounced on multiple GPUs with higher computational accuracy and speed.

Figures 11(a) and 11(b) show the effects of different components on PRC curves. The symbol "+" indicates the number of design components the model has. As Figure 11(a) illustrates, the performances of models when detecting the leading edge's reconstruction areas were enhanced by introducing the FcaNet and MetaAcon. The imported components' effects on detecting the reconstruction area of the trailing edge were unremarkable. Figure 12 is the convergence curves of the models with different components. The training epochs of the Deformable DETR whose backbone was replaced by FcaNet were the same as Deformable DETR with ResNet with an increase of mAP. After importing the MetaAcon function, the training epochs



(a) Raw image (b) Detection results of Deformable DETR

(c) Detection results of LETR (d) Detection results of GLETR

Figure 17: Comparison of detection results of the bottom area.

went to 92. The ghost modules lowered the training epochs significantly to less than 40 and the decrease in mAP was tolerable.

Table 3 presents ablations for three design components of LETR. If we use ResNet as the backbone, LETR degenerates to Deformable DETR, and the mAP is 90.6%. Using FcaNet instead of ResNet improves mAP by 0.6%. Next, replacing ReLU with the meta-Acon activation function can effectively improve mAP by 0.7%. By importing the ghost module, the weight of LETR dropped to 424.4 MB, and the mAP of GLETR is 91.8%, which was lower than that of LETR. This decrease is acceptable. More importantly, the training hour is lowered by the ghost module from 8.5 hours to 3.5 hours. Therefore, it is proved that GLETR performs as good as LETR on the LDEG2021 dataset with fewer parameters and a faster speed of convergence, even though there was a tiny decrease in mAP.

We need to point out that the edge shapes of reconstructed leading/trailing edges are different. Moreover, the reconstructed leading edges' shapes vary from the bottom to the top of the blade. In the rest part of this section, we will discuss the performance of LETR and GLETR when detecting the reconstruction areas of leading/trailing edges and the reconstruction areas of leading edges from different heights of the blade.

The detection results on different reconstructed trailing edges are demonstrated in Figures 13 and 14. In Figure 13, the baseline model, Deformable DETR, detected all targets while some detected objects were misclassified. On the contrary, LETR detected all reconstruction areas of trailing edges without any misclassification as shown in Figure 13(c). However, GLETR missed two targets even though the detected targets were correctly recognized.

We also showed an example in Figure 14 where Deformable DETR failed to detect all reconstruction areas of trailing edges and recognized the wrong targets. By comparing Figures 14(a) and 14(b), we knew that only one target was detected successfully. It was unexpected that Deformable DETR classified some interferences in the background as reconstruction areas of trailing edges. Figures 14(c) and 14(d) give the detection results of LETR and GLETR on reconstructed trailing edges.

To test LETR and GLETR's performance in detecting the reconstruction areas of leading edges of the near-net-shaped blades, we picked raw images from the bottom, middle, and top areas of the blade in LDEG2021 and tested three models on them. The detection results on the top area's images of the blade are shown in Figure 14. The reconstructed curves of the blade's top area usually contain limited reconstruction areas. That is, the targets are relatively smaller. Figure 15(a) is the raw image of this area's reconstructed curves. As Figures 15(b)–15(d) show, Deformable DETR did not perform well while LETR and GLETR recognized these small reconstruction areas with high probability. LETR and GLETR detected all four targets with no mistakes.

As for the middle area of the blade, the reconstructed curves of this area retained more reconstruction areas, as shown in Figure 16(a). The detection results of the middle area of the blade are given in Figures 16(b)–16(d). All three models performed well with no reconstruction area missed.

The detection results of the bottom area of the blade are seen in Figure 17. Figure 17(a) indicates that the cross sections to be further machined in this area are quite dense, which means that the reconstruction areas may be obscured by each other. The test results in Figures 17(b) and 17(c) illustrate that Deformable DETR missed two targets whilst LETR missed only one target. However, GLETR did not behave well, which disregarded five small reconstruction areas in this area.

From Figures 13–17, it also can be summarized that the classification probability of LETR is higher than that of Deformable DETR in most cases. On the anther hand, the classification probability of GLETR is lower than that of LETR and Deformable DETR as a result of importing ghost modules.

The experiment results on the reconstruction areas of leading and trailing edges demonstrate that LETR and GLETR have superior performance. Nonetheless, GLETR's performance in detecting dense and small targets still needs to be improved, which is one of the focuses of our future works.

To test the performance of LETR and GLETR on general small objects detection tasks, we also conducted experiments on two remote sensing datasets: RSOD [60] and NWPU VHR-10 [61]. However, the results on these two remote sensing datasets were not ideal. Due to the imbalanced numbers of different classes' images, we encountered the long-tail effect. We plan to address this problem by applying more data augmentation methods, not just random flipping, cropping, and resizing.

We also want to mention that this paper still has some subsequent works to be accomplished. The reconstruction areas' sizes in images are quite different from their actual sizes. Hence, the correspondence between the actual sizes and the sizes in an image is of great essence. In this article, we did not give such correspondence. So, it is worth exploring how to transfer the detected positions of reconstruction areas in images into real-world models.

6. Conclusions

Aiming to detect the reconstruction area of the near-net-shaped blade, this paper proposed two end-to-end and anchor-free models based on Deformable DETR. Experiment results show that the proposed models have higher accuracy and less weight, respectively. They also offer strong support to our following works. The main contributions of this article are concluded as follows:

(1) We optimized the architecture of Deformable DETR from the aspect of feature extraction and activation function. The new model was named after LETR. LETR extracts features from a frequency-channel mixed domain and activates nonlinear units dynamically. The test results on the self-made dataset LDEG2021 surpassed the baseline model by mAP of 1.3% on a single GPU

(2) On the other hand, we imported the ghost module to LETR and presented a lightweight model, GLETR. Compared with LETR, GLETR achieved a faster convergence speed and less model weight with a tiny decrease in accuracy on LDEG2021. It is proved that GLETR has the potential to be applied to real-time detection

(3) We are capable of obtaining the position and area of the reconstruction area with high efficiency and no errors caused by human experience by applying LETR and GLETR to our task. With the position and area, we can know the geometric parameters of the reconstructed curves by exploring parameter extraction algorithms

LETR and GLETR are two successful attempts combining object detection and adaptive machining. However, there are some issues remained in this paper and can be investigated in the future:

(1) GLETR has its disadvantages when detecting the targets in the bottom area of the blade. That is due to the fact that the objects cover each other and some features are ignored by the ghost module. One potential solution to this problem is to feed the network with images of larger sizes during the training process, by which the position and semantic information is enhanced

(2) The test results of LETR and GLETR on remote sensing datasets were unsatisfactory due to the long-tail effect. Here, we suggest adopting more data augmentation algorithms to keep the number of objects' classes balanced. A promising way is to stitch the images, especially for the objects which have fewer images

(3) We need to extract the geometric parameters in images in the future. In detail, the next step is to find the relationship between the coordinates of the reconstruction area's each pixel and its corresponding geometric parameters, like curvature and chord length

Data Availability

The data of this article is within the paper. The code and dataset will be available after acceptance.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Acknowledgments

This work is supported by the National Science and Technology Major Project of China (J2019-VII-0001-0141), the National Nature Science Foundation of China (51905441), and the Fundamental Research Funds for Central Universities of China (31020200506002). The authors sincerely thank all anonymous editors and reviewers for their constructive review of this manuscript. The preprint of this paper has been published [62].

References

[1] J. E. Makem, O. Hengan, and C. G. Armstrong, “A virtual inspection framework for precision manufacturing of aerofoil components,” *Computer-Aided Design*, vol. 44, no. 9, pp. 858–874, 2012.

[2] Y. Feng, J. Ren, and Y. Liang, “Prediction and reconstruction of edge shape in adaptive machining of precision forged blade,” *International Journal of Advanced Manufacturing Technology*, vol. 96, no. 5-8, pp. 2355–2366, 2018.

[3] Z. Yin, J. Ren, and Y. Liang, “Classification of blade’s leading edge based on neural networks in adaptive machining of near-net-shaped blade,” *International Journal of Precision Engineering and Manufacturing*, vol. 22, no. 11, pp. 1817–1828, 2021.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., “Generative adversarial nets,” *Advances in Neural Information Processing Systems*, vol. 27, p. 16, 2014.

[5] L. Liu, W. Ouyang, X. Wang et al., “Deep learning for generic object detection: a survey,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, 2020.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Columbus, Ohio, USA, 2014.

[7] R. Girshick, “Fast R-CNN,” in *2015 IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[10] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6517–6525, 2017.

[11] J. Redmon and A. Farhadi, “YOLOv3: an incremental improvement,” 2018, http://arxiv.org/abs/1804.02767v1.

[12] A. Bochkovskiy, C. Y. Wang, and H. Liao, “YOLOv4: optimal speed and accuracy of object detection,” 2020, http://arxiv.org/abs/2004.10934v1.

[13] W. Liu, D. Anguelov, D. Erhan et al., “SSD: single shot multibox detector,” in *European Conference on Computer Vision*, vol. 9905, 2016.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[15] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: deformable transformers for end-to-end object detection,” 2020, http://arxiv.org/abs/2010.04159v2.

[16] Z. Yun, C. Zhi-Tong, and N. Tao, “Reverse modeling strategy of aero-engine blade based on design intent,” *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9-12, pp. 1781–1796, 2015.

[17] Z. Zhao, Y. Fu, X. Liu, J. Xu, J. Wang, and S. Mao, “Measurement-based geometric reconstruction for milling turbine blade using free-form deformation,” *Measurement*, vol. 101, pp. 19–27, 2017.

[18] H. Yu, L. Xuegeng, and P. Liu, “Stream surface reconstruction of aero engine blade based on limited measured points,” *Advances in Engineering Software*, vol. 131, pp. 90–101, 2019.

[19] Y. Zhang, Z. Chen, and Z. Zhu, “Adaptive machining framework for the leading/trailing edge of near-net-shape integrated impeller,” *International Journal of Advanced Manufacturing Technology*, vol. 107, no. 9-10, pp. 4221–4229, 2020.

[20] D. Wu, H. Wang, K. Zhang, B. Zhao, and X. Lin, “Research on adaptive CNC machining arithmetic and process for near-net-shaped jet engine blade,” *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 717–744, 2020.

[21] B. Singh and L. S. Davis, “An analysis of scale invariance in object detection - SNIP,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3578–3587, 2018.

[22] B. Singh, M. Najibi, and L. S. Davis, “SNIPER: efficient multi-scale training,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] W. Zhang, L. Jiao, X. Liu, and J. Liu, “Multi-scale feature fusion network for object detection in VHR optical remote sensing images,” in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 330–333, 2019.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 936–944, 2017.

[25] W. Liu, L. Ma, and C. He, “Arbitrary-oriented ship detection framework in optical remote-sensing images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 6, pp. 937–941, 2018.

[26] H. Law and J. Deng, “CornerNet: detecting objects as paired keypoints,” *International Journal of Computer Vision*, vol. 128, no. 3, pp. 642–656, 2018.

[27] X. Zhou, J. Zhuo, and P. Krähenbühl, “Bottom-up object detection by grouping extreme and center points,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 850–859, 2019.

[28] X. Zhou, D. Wang, and P. Krhenbühl, “Objects as points,” 2019, http://arxiv.org/abs/1904.07850v2.

[29] X. Zhou, V. Koltun, and P. Krhenbühl, “Probabilistic two-stage detection,” 2021, http://arxiv.org/abs/2103.07461v1.

[30] R. Dong, D. Xu, J. Zhao, L. Jiao, and J. An, “Sig-NMS-based faster R-CNN combining transfer learning for small target detection in VHR optical remote sensing imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 11, pp. 8534–8545, 2019.

[31] Z. Cai and N. Vasconcelos, “Cascade R-CNN: delving into high quality object detection,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[32] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing [review article],” *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[33] A. Vaswani, N. Shazeer, N. Parmar et al., “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[34] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision*, pp. 213–229, 2020.

[35] Z. Sun, S. Cao, Y. Yang, and K. M. Kitani, "Rethinking transformer-based set prediction for object detection," 2020, http://arxiv.org/abs/2011.10881v1.

[36] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019.

[37] J. Dai, H. Qi, Y. Xiong et al., "Deformable convolutional networks," in *2017 IEEE International Conference on Computer Vision*, pp. 764–773, 2017.

[38] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.

[39] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," 2013, http://arxiv.org/abs/1302.4389v1.

[40] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.

[41] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, http://arxiv.org/abs/1710.05941.

[42] N. Ma, X. Zhang, and J. Sun, "Funnel activation for visual recognition," in *European Conference on Computer Vision*, pp. 351–368, 2020.

[43] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic relu," in *European Conference on Computer Vision*, pp. 351–367, 2020.

[44] N. Ma, X. Zhang, and J. Sun, "Activate or not: learning customized activation," 2020, http://arxiv.org/abs/2009.04759v2.

[45] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "Cnnpack: packing convolutional neural networks in the frequency domain," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[46] C. Liu, Y. Wang, K. Han, C. Xu, and C. Xu, "Learning instance-wise sparsity for accelerating deep models," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3001–3007, 2019.

[47] B. Jacob, S. Kligys, B. Chen et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

[48] H. Chen, Y. Wang, C. Xu et al., "Data-free learning of student networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.

[49] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: more features from cheap operations," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1577–1586, 2020.

[50] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[51] A. G. Howard, M. Zhu, B. Chen et al., "MobileNets: efficient convolutional neural networks for mobile vision applications," 2017, http://arxiv.org/abs/1704.04861v1.

[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[53] A. Howard, M. Sandler, G. Chu et al., "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

[54] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[55] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: an extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[56] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: practical guidelines for efficient CNN architecture design," in *European Conference on Computer Vision*, pp. 122–138, 2018.

[57] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," in *European Conference on Computer Vision*, pp. 740–755, 2014.

[58] Z. Qin, P. Zhang, F. Wu, and X. Li, "FcaNet: frequency channel attention networks," 2020, http://arxiv.org/abs/2012.11879.

[59] K. Chen, J. Wang, J. Pang et al., "MMDetection: open MMLab detection toolbox and benchmark," 2019, http://arxiv.org/abs/1906.07155.

[60] Y. Long, Y. Gong, Z. Xiao, and Q. Liu, "Accurate object localization in remote sensing images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 5, pp. 2486–2498, 2017.

[61] C. Gong, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, 2016.

[62] Z. Yin, Y. Liang, J. Ren, J. An, and F. He, "LETR: an end-to-end detector of reconstruction area in blade's adaptive machining with transformer," 2022, Preprints: 2021090332.