LETR: An End-to-End Detector of Reconstruction Area in Blade's Adaptive Machining with Transformer

Zikai Yin[a], Yongshou Liang ✉[a], Junxue Ren[a], and Jungang An[b]

[a]School of Mechanical Engineering, Northwestern Polytechnical University
yinzikai1997@mail.nwpu.edu.cn, liangyongshou@nwpu.edu.cn, rjx1968@nwpu.edu.cn
[b]Haimo Research Institution
jungangan001@gmail.com

August 2021

Abstract

In the leading/trailing edge's adaptive machining of the near-net-shaped blade, a small portion of the theoretical part is retained for securing aerodynamic performance by manual work. However, this procedure is time-consuming and depends on the human experience. In this paper, we defined retained theoretical leading/trailing edge as the reconstruction area. To accelerate the reconstruction process, an anchor-free neural network model based on Transformer was proposed, named LETR (Leading/trailing Edge Transformer). 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 GLETR (Ghost Leading/trailing Edge Transformer) 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.

Keywords: Near-net-shaped Blade; Adaptive Machining; Small Object Detection; Neural Network; Transformer; Real-Time Detection

1 Introduction

The near-net-shaped blades are applied to the blades of aero engine as it fits the modern aero engine performance better. The blank material and typical structure of near-net-shaped blade are shown in Fig. 1(a) and Fig. 1(b) respectively. A section curve of the blade is presented in Fig. 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. That means we cannot plan the tool path according to the designed model. Hence, we need to reconstruct the theoretical leading /trailing edge. 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 Fig. 3. This process, however, is time-consuming and depends on the human experience. In this case, adaptive machining 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.

Unlike traditional ways based on geometric prediction, we reconstruct models from a brand new perspective. Our method can be described in the following steps. In the first stage, we use Generative Adversarial Networks (GAN) [1] to generate optimal reconstructed leading/trailing edges' images based on our previous manual works. Next, we need to detect the retained part and bridged curves of these reconstructed curves. Thirdly, we will extract the parameter information of them. As long as these parameters are obtained, we can utilize computer-aided design (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 purpose is to detect the retained part of the theoretical leading/trailing edge and we use the item 'reconstruction area' to represent them.

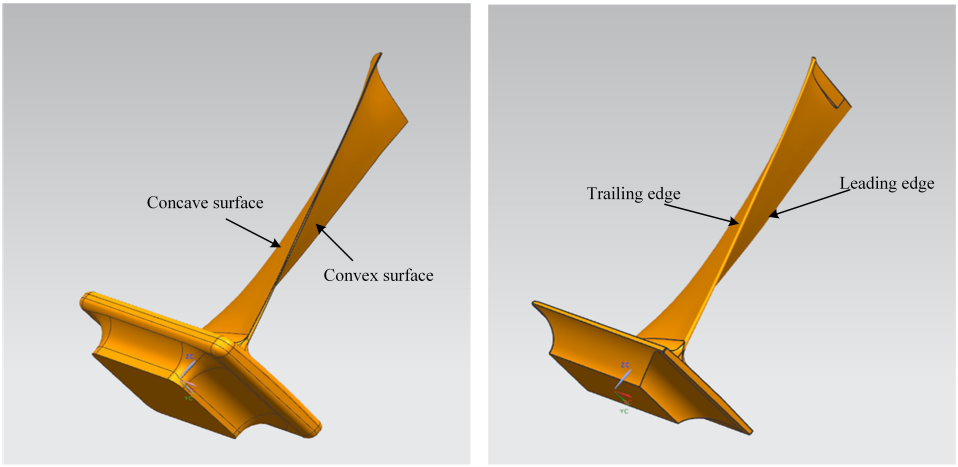


Figure 1: The typical structure of near-net-shaped blade. (a)The blank to be formed. (b) The near-net-shaped blade.

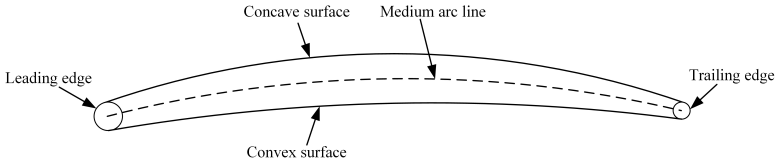


Figure 2: A section line of the near-net-shaped blade

A small object usually contains a small quantity of semantic information due to its small size. As shown in Fig. 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) [2], 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 RCNN series[3, 4, 5], 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[6, 7, 8, 9] and Single Shot Multi-Box Detector (SSD) [10].

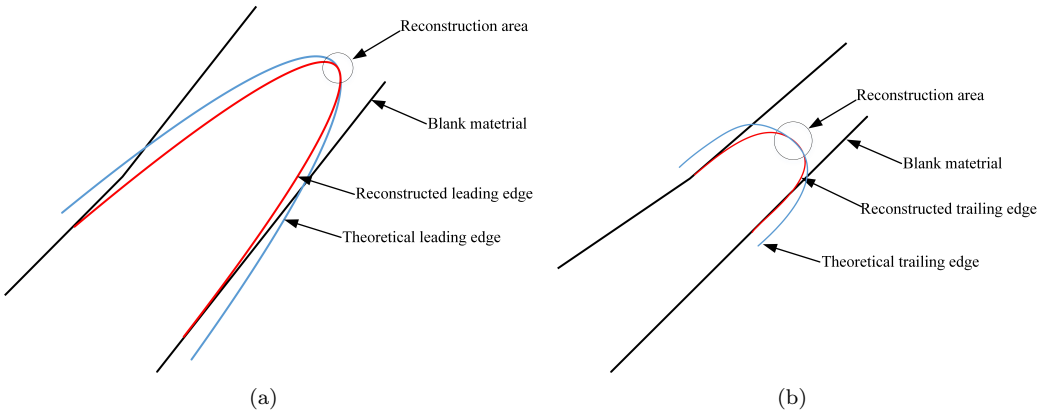


Figure 3: The reconstruction area of a blade. (a) Leading edge's reconstruction area. (b) Trailing edge's reconstruction area.

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[11]. Moreover, the NMS algorithm isn't sensitive to small objects which contain less semantic information. The number of network's predictions is much larger than that of actual objects' number and NMS is employed here to remove superfluous predicted bounding boxes. The NMS sets a threshold value and deletes all bounding boxes higher than the threshold value and keeps the bounding box with the highest score. The post-process 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 post-processing of the NMS algorithm. In recent years, Transformer has shown its superiority in the object detection field due to its capacity of spatial relation modeling of targets. The model that works best now with Transformer is Deformable DETR[12]. 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 firstly proposed an anchor-free model named LETR (Leading/ trailing Edge Detection Transformer). LETR extracts features of reconstruction areas by using a combination of channel attention and frequency analysis. Besides, LETR activates non-linear 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 ghost modules. The model is called GLETR (Ghost Leading/ trailing Edge Detection Transformer).
3. We tested LETR and GLETR on the self-made dataset LDEG2021. It is proved that LETR outperformed the baseline model and showed state-of-the-art performance on detecting reconstruction areas. Moreover, GELTR has significantly fewer parameters and converges faster than LETR in the training process with a higher mAP.

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 final conclusions and future works.

2 Related works

2.1 Reconstruction of edge shape

The prevalent model reconstruction methods can be concluded in three aspects: by fitting curves of measured data, by predicting the deformation of the blade's surface, and by 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 edge shape procedure. Some representative researches are illustrated as follows. Yun Zhou, et.al[13] imported parameterization design to Free-Form Deformation and realized the model reconstruction. Zhao et. al [14] improves the accuracy of reconstruction by inserting knots. Feng, et al.[15] proposed a fairing algorithm while preserving the design intent. Yu, et.al[16] established the relationship of measured points and velocity field of the blade's section. Further, they calculated the curves in the area without measuring data. Zhang, et.al[17] considered the constraints of the chord length, angle of attack, and radius during the reconstruction process. Wu, et al[18] proposed a novelalgorithm to removed bad measuring points.

Nevertheless, the current researches on modeling of large curvature inexact forming regions mainly focus on simple geometric element fitting but do not fully consider the relationship of the design intention of the blade, the similarity relationship between 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. For this article, we furtherly 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 fewer 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[5]. Bharat Singh, et.al[19] proposed Scale Normalization for Image Pyramids (SNIP) for small object detection. Furtherly, they modified SNIP and proposed SNIPER[20] for efficient and fast detection. Zhang, et.al[21] utilized a multi-scale feature fusion layer (MFL) and a certain extent of improvement was obtained. On the other hand, a typical one-stage detector is RetinaNet[11], which uses focal loss and feature pyramid maps[22] to detect small objects. Based on YOLO v2, the work of Liu, et.al[23] can detect arbitrary-oriented targets of small sizes.

Some researchers proposed anchor-free methods. The representative works include CornerNet[24], ExtremeNet[25], and CenterNets[26, 27]. 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[28] integrated transfer learning with Faster RCNN for annotation and improved precision. Cai, et.al[29] 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 post-processing. 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 progressing of natural language processing[30] (NLP) has led researchers to investigate its application in computer vision. Vaswani, et.al[31] introduced Transformer in NLP for the first time. Transformer achieves impressive results on sequence prediction. Carino, et al.[32] 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 doesn't perform well on small object detection tasks. Sun, et.al[33] argued the cross-attention module of DETR is not necessary and proposed two models named TSP-FCOS and TSP-RCNN respectively. Zhu, et. al[12] imported deformable convolution[34] to DETR and proposed Deformable DETR. Deformable DETR deals with input feature maps from the spatial domain. 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 functions are ReLU[35] and its variants such as Leaky ReLU[36], SiLU[37], and DY-ReLU[38]. When implemented on deep neuron networks, however, the abovementioned activation functions' effects become weaker. Unlike previous activation functions, Acon[39] functions learn whether to activate the specific neurons and convert models into dynamic networks, which brings better performance as networks going 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. Some typical models are proposed by researchers. MobileNet series[40, 41, 42], for example, are based on depth-wise and point-wise separable convolutions. Another famous light-weighted models are ShuffleNet[43, 44] series, which exchange their inner channel information to reduce computational cost. The redundancy of feature maps has not been solved well. GhostNet[45], 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[46]. In this part, we briefly reviewed the inner mechanism of DETR.

3.1.1 Multi-head attention

In the natural language processing field, the self-attention mechanism was adopted in the Transformer model. By computing the attention value of a given query set (target words) and a key set (source words), the Trans-

former modulates the compatibility of queries and keys. In the multi-head attention mechanism, the outputs of several attention heads are aggregated linearly with learnable weight parameters. The formulation of multi-head attention is shown as:

$$MultiHeadAttn(z_q, x) = \sum_{m=1}^{M} W_m[\sum_{k\in\Omega_k} A_{mqk} \cdot W_m^{'} x_k] \tag{1}$$

3.1.2 Position embedding

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

$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{2}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{3}$$

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 model 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:

$$\sigma = \arg\min_{\sigma\in P_N} \sum_{i}^{N} L_{match}(y_i, y_{\sigma(i)}) \tag{4}$$

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

$$L_{match} = \sum_{i=1}^{N}[-logP_{\sigma(i)} + L_{box}(b_i, \hat{b_i})] \tag{5}$$

Albeit the fact that DETR has achieved state-of-the-art performance, it costs massive computational resources and itsconvergence speed is relatively low. Furthermore, the performance of DETR on small target datasets is disappointing. Besides, the test results of DETR presents no outstanding performance on our detection task. Hence, it is proved that DETR is not suitable for our detection task.

3.1.4 Deformable attention

DETR didn't perform well on small target detection tasks, owing to its inner mechanism. Zhu et al. [12] combined DETR with deformable convolution and named their model as 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:

$$DeformAttn(z_q, p_q, x) = \sum_{m=1}^{M} W_m[\sum_{k=1}^{K} A_{mqk} \cdot W_m^{'} x(p_q + \Delta p_{mqk})] \tag{6}$$

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 multi-scale size of input features, the deformable attention generates to multi-scale deformable attention and can be calculated by:

$$MSDeformAttn(z_q, \hat{p_q}, x^l\ |_{l=1}^{L}) = \sum_{m=1}^{M} W_m[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlpk} \cdot W_m^{'} x^l((\phi_q) + \Delta p_{mlqk})] \tag{7}$$

where 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 Fig. 4, our model adopted state-of-the-art methods implemented in small object detection. A channel-frequency mixed backbone 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 [12].

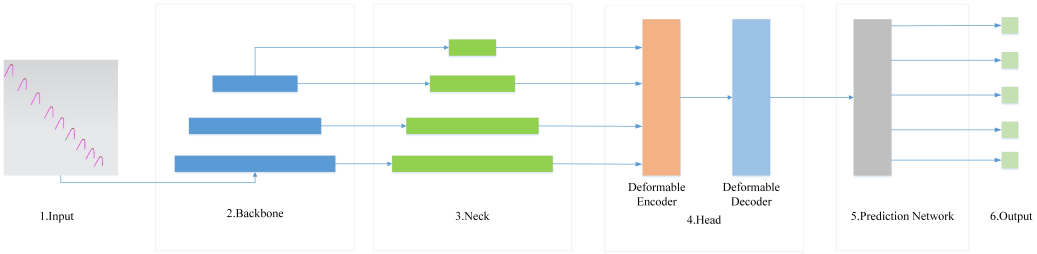


Figure 4: Overview of LETR

4.2 Feature extraction

Technically, the backbone is responsible for extracting feature maps over input images. Previous 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 multi-scale object detection. Furthermore, inspired by FcaNet[47], we imported the multi-spectral channel attention module of FcaNet to the feature extraction process of the model's backbone. The details of the utilized module are demonstrated in Fig. 5.

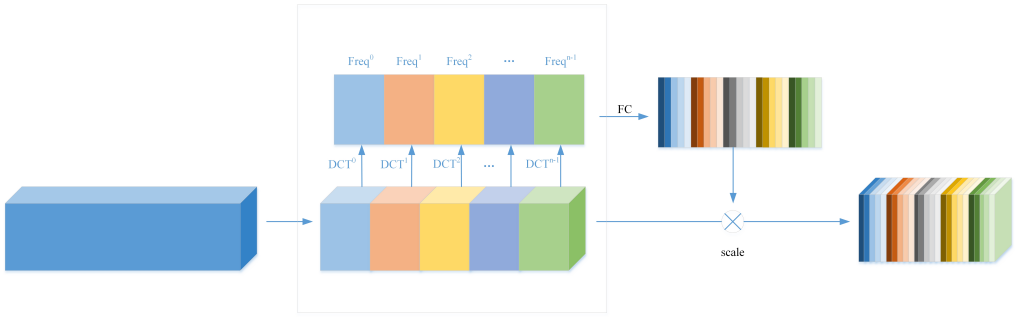


Figure 5: The feature extraction process of the multi-spectral channel attention module.

Assuming that the dimension of the input feature is $C \times H \times W$, the multi-spectral 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 Eq. (8):

$$2DDCT(x^{2d}) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} x_{i,j}^{2d} cos(\frac{\pi h}{H}(i+\frac{1}{2}))cos(\frac{\pi w}{W}(j+\frac{1}{2})) \tag{8}$$

where $h \in [0, 1, \cdots, H-1]$, $w \in [0, 1, \cdots, W-1]$.

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:

$$X = [X^0, X^1, \cdots, X^{n-1}] \tag{9}$$

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

$$Freq^i = 2DDCT(X^i) \tag{10}$$

Thus, by concatenating all $Freq_i$, we have the final frequency feature $Freq$ of the input feature:

$$Freq = cat([Freq^0, Freq^1, \cdots, Freq^{n-1}]) \tag{11}$$

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

$$MSAttn = sigmoid(fc(Freq)) \tag{12}$$

where $sigmoid$ is sigmoid function and $fc(\cdot)$ is a fully connected layer. By rescaling X with MSAttn , the output of multi-spectral channel attention module, denoted as $\widetilde{X}$, is computed as:

$$\widetilde{X} = F_{scale}(X, MSAttn) = MSAttn \times X \tag{13}$$

The mixed frequency and channel weights are applied to the input feature maps after the multi-spectral 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 neuron network's architecture, activation function plays a key role in importing non-linearity to improve the model's classification capability. Considering the big difference of foreground and background of our task, we employed Acon functions because of their excellent ability to control the extent of non-linear 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 the special cases of Acon-C. According to Ma et.al[39], 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 $max(x_1, x_2, \cdots, x_n)$:

$$s_\beta(x_1, x_2, \cdots, x_n) = \frac{\sum_{i=1}^{n} x_i e^{\beta x_i}}{\sum_{i=1}^{n} e^{\beta x_i}} \tag{14}$$

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, $s_\beta(x_1, x_2, \cdots, x_n)$ degenerizes to the following format:

$$s_\beta(\eta_a(x), \eta_b(x)) = (\eta_a(x) - \eta_b(x)) \cdot \sigma[\beta\eta_a(x) - \eta_b(x)] + \eta_b(x) \tag{15}$$

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 $\eta_a(x) = p_2 x$, Acon-C is written as:

$$AconC(x) = s_\beta(p_1 x, p_2 x) = (p_1 - p_2)x \cdot \sigma[\beta(p_1 - p_2)x] + p_2 x \tag{16}$$

And furtherly, we see β as a learnable network $G(x)$ and $AconC(x)$ can be generated to $MetaAconC(x)$, which is computed as:

$$MetaAconC(x) = (p_1 - p_2)x \cdot \sigma[G(x)(p_1 - p_2)] + p_2 x \tag{17}$$

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

4.4 Ghost module

As Fig. 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 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.

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

$$Y = X * f + b \tag{18}$$

where $*$ and b denote the convolution operation and bias respectively, 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:

$$Y^{'} = X * f^{'} \tag{19}$$

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 ghost module are written as:

$$y_{ij} = \phi_{i,j}(y_i^{'}) \tag{20}$$

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 Fig. 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 channel dimension. In practice, the inner connection of the ghost module is shown in Fig. 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 FcaAconNet backbone with ghost module Ghost-FcaAconNet.

5 Experiments

In this part, we conducted comparison experiments on the self-made dataset LDEG2021 to show that LETR and GLETR have competitive performance on the leading/trailing edge detection task. The code will be available at https: //github.com/andrewsilver1997/LETR.

5.1 Dataset

We made LDEG2021 to detect the reconstruction area of the leading and trailing edge. The whole images of LDEG2021 are section curves of different reconstruction areas of leading and trailing edges. LDEG2021 has 397 images annotated with two classes: leading edge and trailing edge. Some selected samples in LDEG2021 are shown in Fig. 7.

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[48] with Pytorch 1.5.1, cuDNN 7.6.1, and CUDA 10.1 for implementation.

5.3 Results and discussions

5.3.1 Evaluation metrics

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

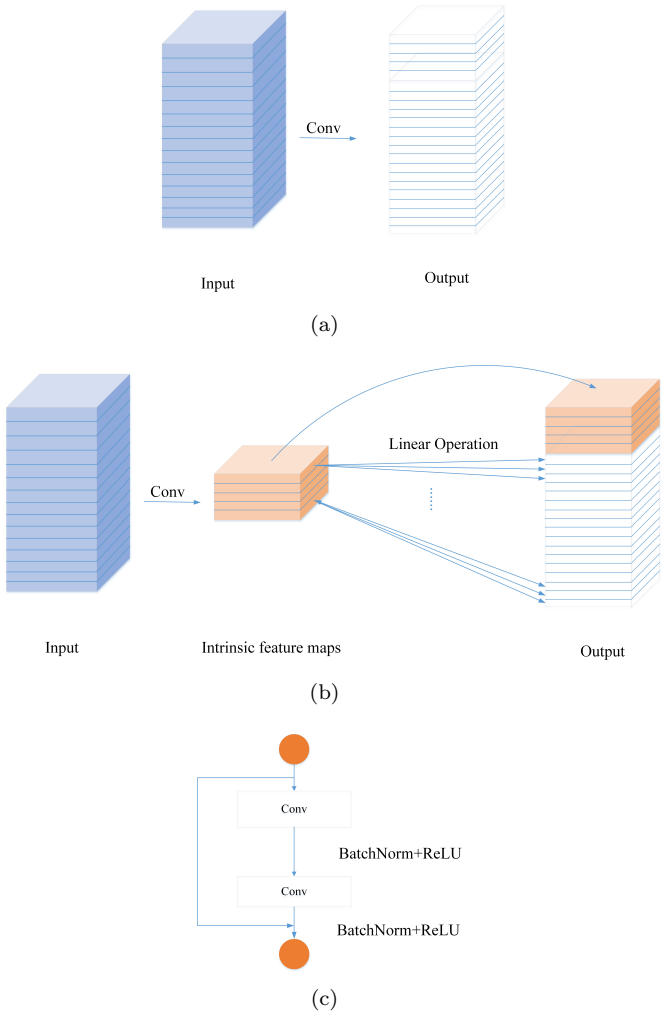


Figure 6: The ghost module in Ghost-FcaAconNet. (a) The convolution operation. (b) The processing of ghost module. (c) The inner connection of the ghost module.

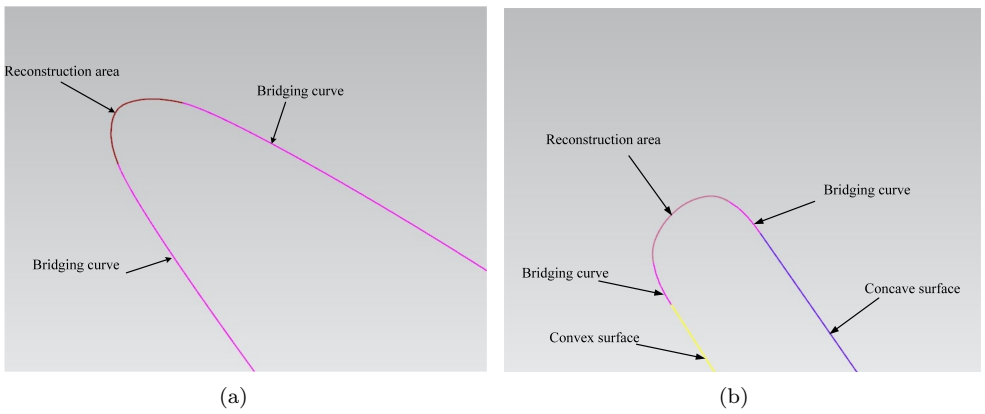


Figure 7: Selected samples in LDEG2021. (a) Leading edge. (b) Trailing edge.

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

Table 1: 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 :

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

$$Recall = \frac{TP}{TP + FN} \tag{22}$$

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

AP (Average Precision) represents the performance that the model detects a specific class of all objects. AP is defined in Eq. (23):

$$AP = \int_0^1 p(r)dr \tag{23}$$

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 Eq.(24).

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{24}$$

where AP_i is the AP value of i^th class, 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.

5.4 Results on LDEG2021 dataset

Fig. 8(a) and (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 ResNer101 were nearly the same when detecting the reconstruction areas of the leading edge. Maintaining the same 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 Fig. 8(b). The convergence curves of the three models are presented in Fig. 9(in detail. 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 Deformable DETR with ResNet50. On the other hand, LETR, achievement 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%.

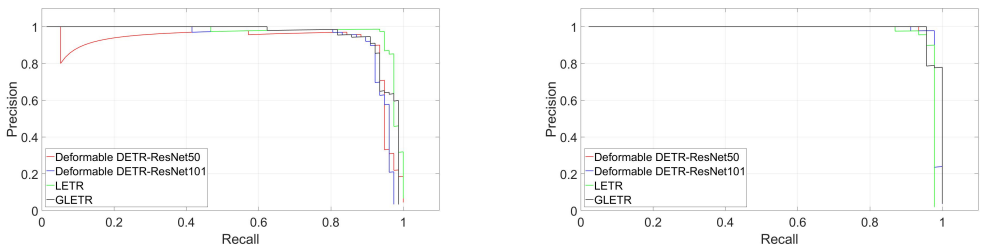


Figure 8: Comparison of PRCs

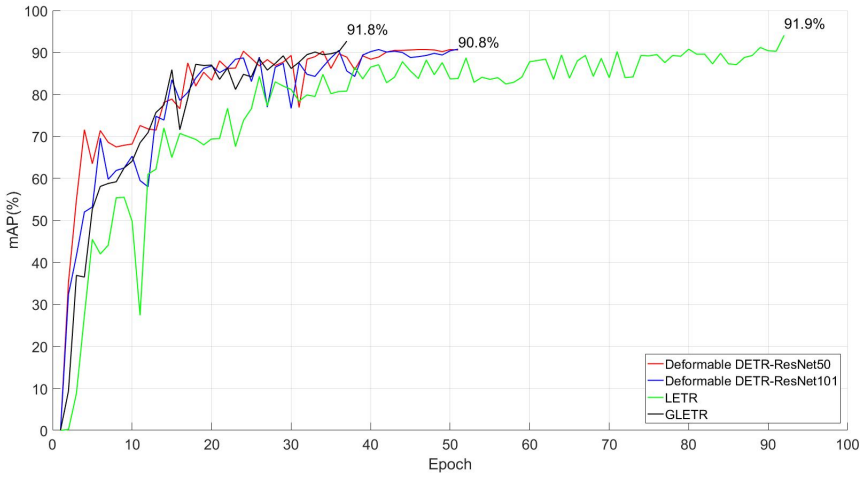


Figure 9: Convergence curves of three models on LDEG2021

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 121.2MB 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 performance of our models will be more pronounced on multiple GPUs with higher computational accuracy and speed.

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

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

Fig. 10(a) and (b) show the effects of different components on PRC curves. The symbol '+' indicates the number of design components the model has. As Fig. 10(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. Fig. 11 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 went to 92. The ghost modules lowered the training epochs significantly to less than 40 and the decrease of mAP was tolerable.

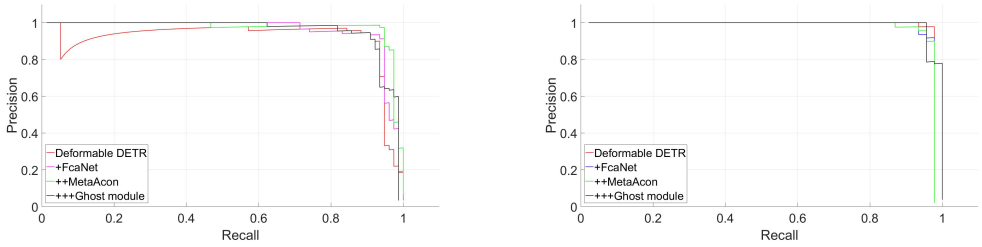


Figure 10: PRCs of the models with different components

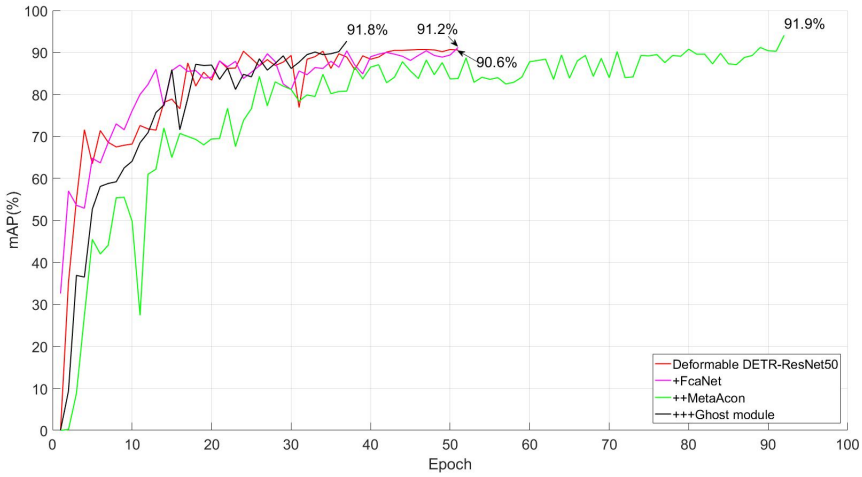


Figure 11: Convergence curves of the models with different components

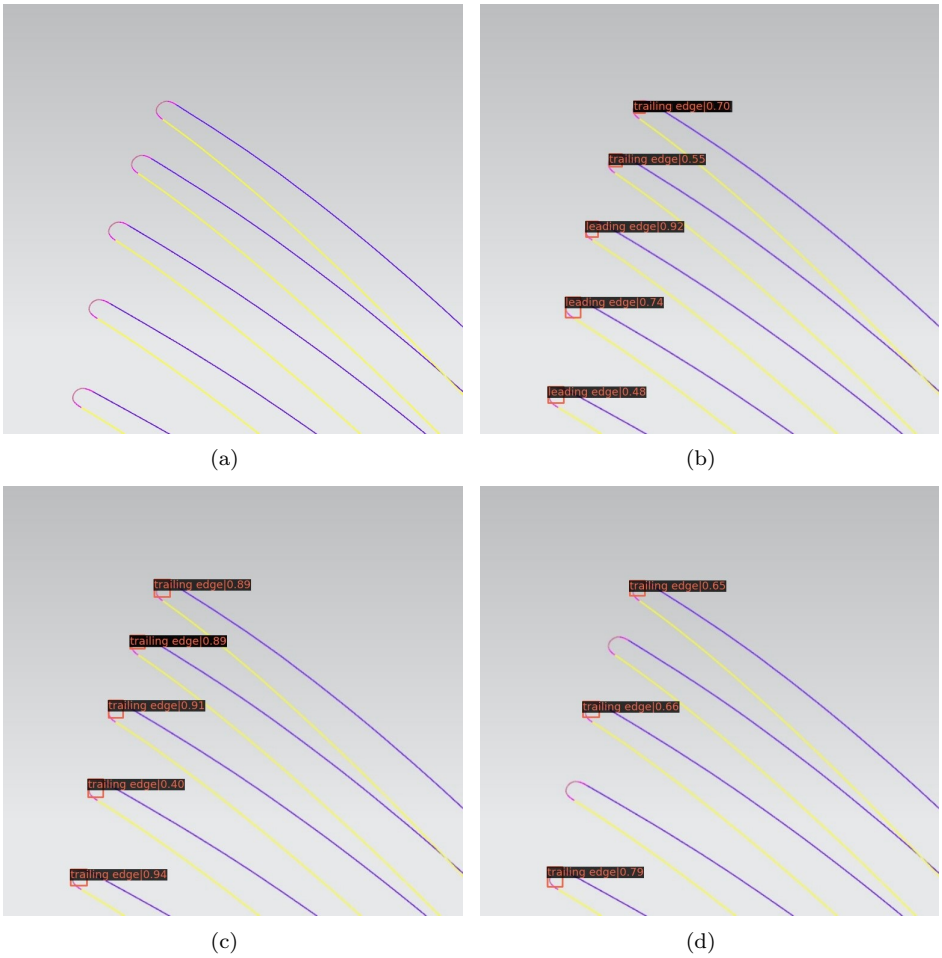
Table 3 presents ablations for three design components of LETR. If we use ResNet as the backbone, LETR degenerates to Deformable DETR and the mAP is 90.6%. Using FcaNet instead of ResNet improves mAP by 0.6%. Next, replacing ReLU with the meta-Acon activation function can effectively improve mAP by 0.7%. By importing the ghost module, the weight of LETR dropped to 424.4MB 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, faster speed of convergence, even though there was a tiny decrease of mAP.

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

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

We need to point out that the edge shapes of reconstructed leading/trailing edges are in a huge difference. Moreover, the reconstructed leading edges' edge shape varies 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 areas of the blade.

The detection results on different reconstructed trailing edges are demonstrated in Fig. 12 and Fig. 13. In Fig.12, 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 Fig. 12(c). However, GLETR missed two targets even though the detected targets were correctly recognized.



(a) (b) (c) (d)

Figure 12: 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.

We also showed an example in Fig. 13 where Deformable DETR failed to detect all reconstruction areas of trailing edges and recognized the wrong targets. By comparing Fig. 13(a) and Fig. 13(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. Fig. 13(c) and Fig. 13(d) give the detection results of LETR and GLETR on reconstructed trailing edges.

To test the performance of LETR and GLETR's performance on 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 Fig. 14. The reconstructed curves of the blade's top area usually contain limited reconstruction areas. That is, the targets are relatively smaller. Fig. 14(a) is the raw image of this area's reconstructed curves. As Fig. 14(b), Fig. 14(c), and Fig. 14(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.

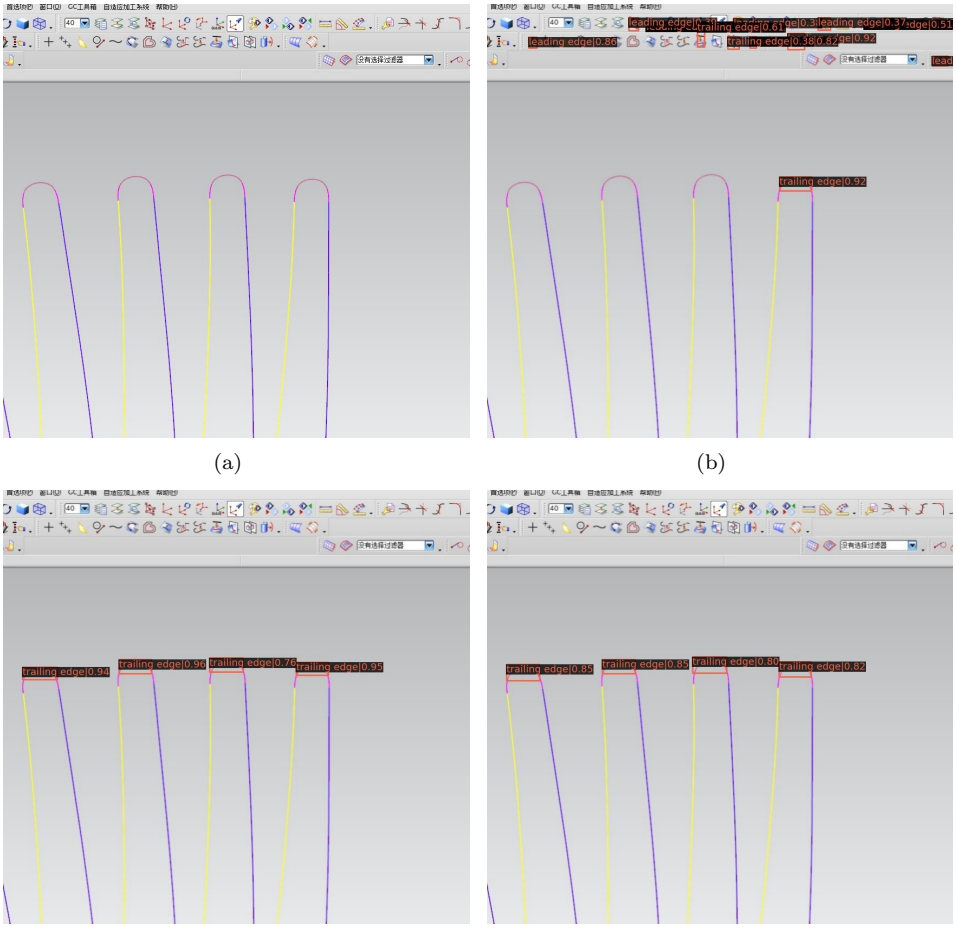
As for the middle area of the blade, the reconstructed curves of this area retained more reconstruction areas, as shown in Fig. 15(a). The detection results of the middle area of the blade are given by Fig. 15(b), Fig. 15(c), and Fig. 15(d). All three models performed well with no reconstruction area missed.

The detection results of the bottom area of the blade are seen in Fig. 16. Fig. 16(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 Fig. 16(b) and Fig. 16(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 Fig.12 to Fig.16, 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.

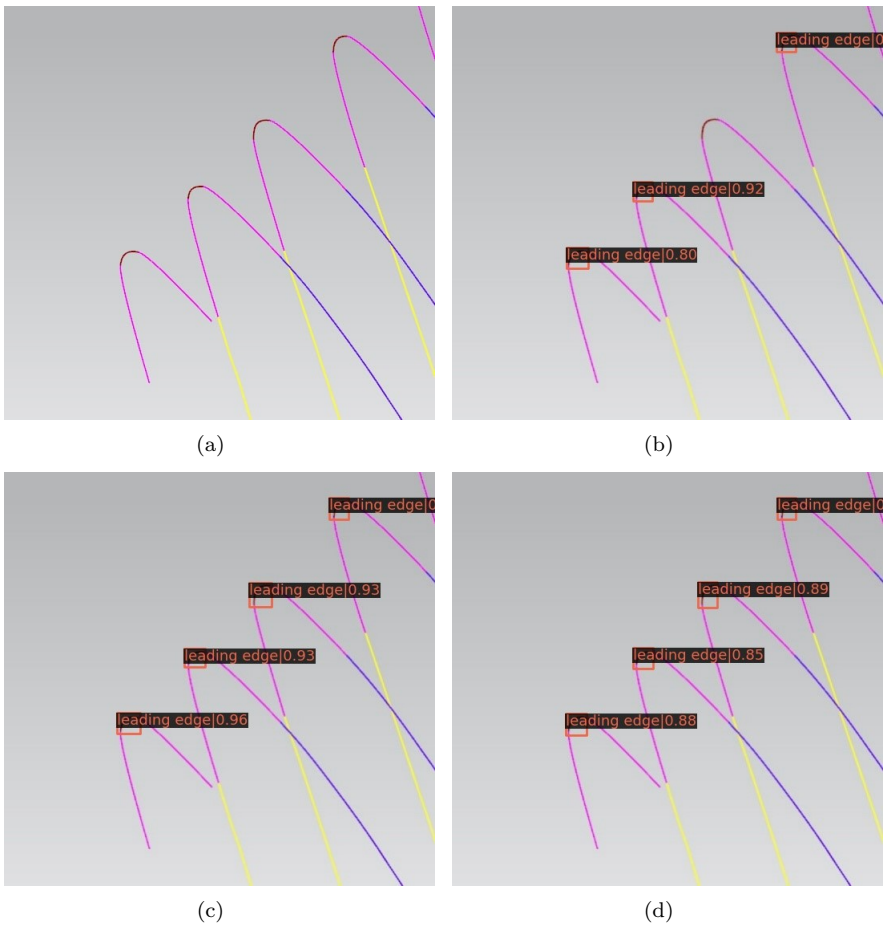
The experiment results on the reconstruction areas of leading and trailing edges demonstrate that LETR and GLETR have superior performance. Nonetheless, GLETR's performance in detecting dense and small targets still needs to be improved.



(a) (b)

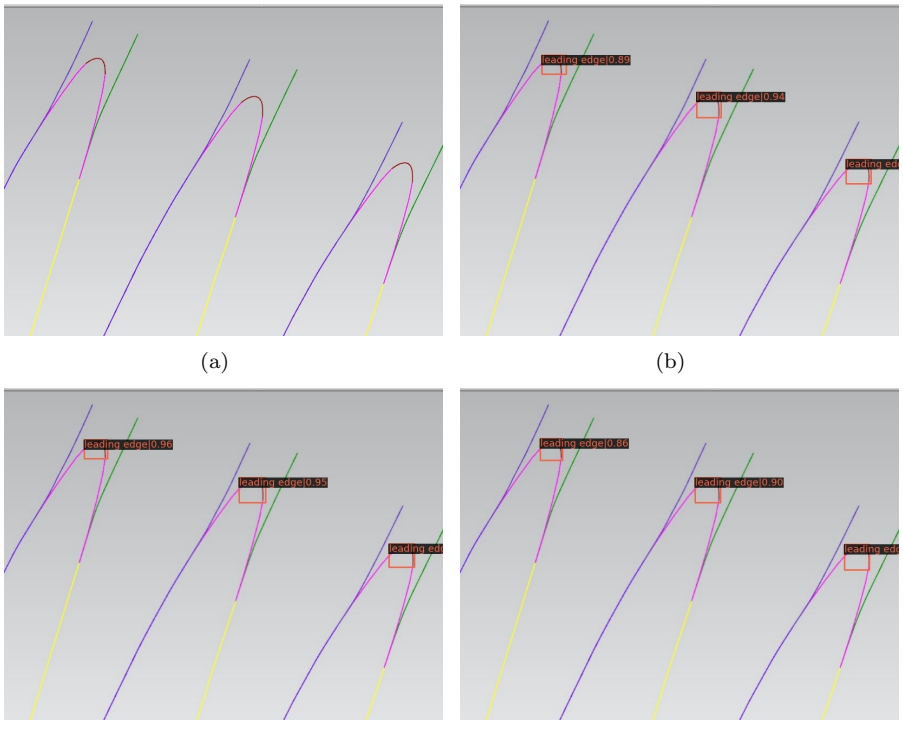
(c) (d)

Figure 13: Another 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.



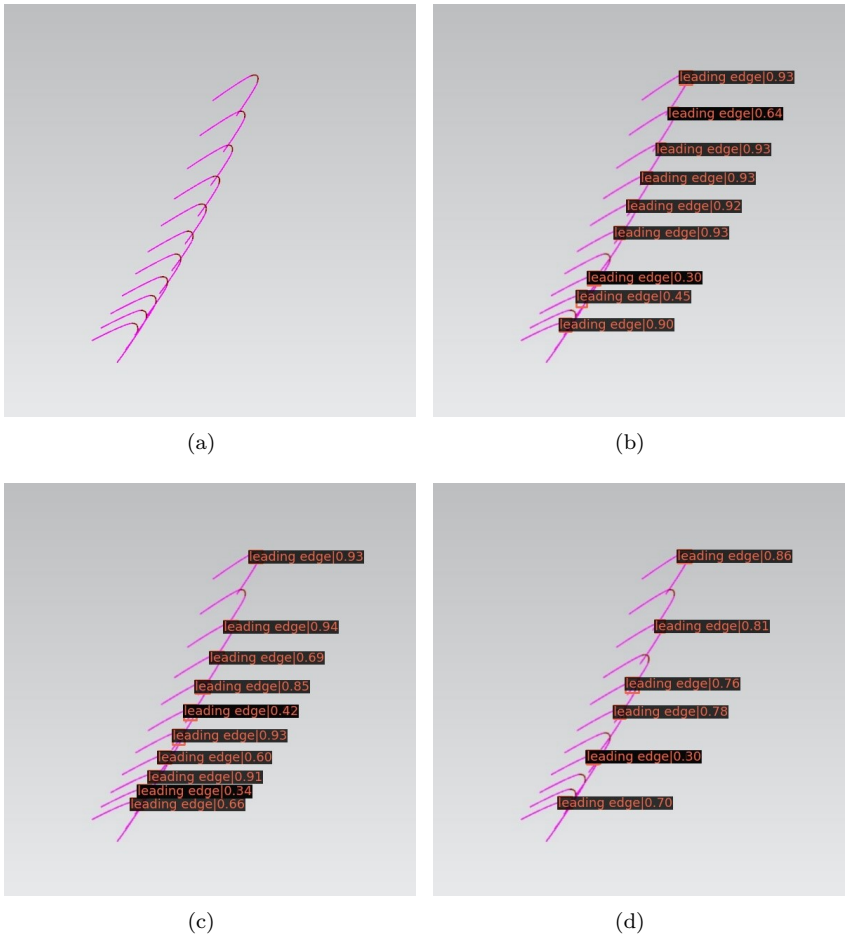
(a) (b) (c) (d)

Figure 14: 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.



(a) (b) (c) (d)

Figure 15: Comparison of detection results of the middle area. (a) Raw image. (b) Detection results of Deformable DETR. (c) Detection results of LETR. (d) Detection results of GLETR.



(a) (b)

(c) (d)

Figure 16: Comparison of detection results of the bottom area. (a) Raw image. (b) Detection results of Deformable DETR. (c) Detection results of LETR. (d) Detection results of GLETR.

6 Conclusions

Unlike the previous reconstruction method of the near-net-shaped blade, we reconstructed blades based on a great number of completed works and deep learning. This paper is focused on the detection of reconstruction areas. In other words, the reconstruction areas are represented by bounding boxes. The main contributions of our article are concluded as follows:

1. Aiming to detect the reconstruction area of the near-net-shaped blade, an end-to-end detector based on Deformable DETR was proposed and named after LETR. LETR extracts features from a frequency-channel mixed domain and activates non-linear 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 increase of mAP on LDEG2021. Experiment results of GLETR prove that it has the potential to be applied to real-time detection.

3. We compared the detection results of different areas and different reconstruction types. It is proved that LETR and GLETR are high-precision and anchor-free models for detecting the reconstruction areas.

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. By importing the ghost modules, the model weight and training time are decreased obviously. But GLETR has its disadvantages when detecting the targets in the bottom area of the blade. The reasons why GLETR did not work well when detecting dense and small targets should be further analyzed.

2. As now we have the bounding box of the reconstruction area, the next step is the modeling of the relationship of the area of bounding boxes and the geometric parameters of reconstructed curves, for example, the function between the bounding boxes' area and the chord length of reconstructed curves.

3. In this article, we only proposed the models for our specific task and did not investigate their performance on general small object detection tasks such as remote sensing and navigation. The generalization performances

of LETR and GLETR deserve further study. We expect more outstanding works in the future.

References

[1] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 27 (NIPS 2014)*. Vol. 27. Advances in Neural Information Processing Systems. 28th Conference on Neural Information Processing Systems (NIPS), Montreal, CANADA, DEC 08-13, 2014. 2014, pp. 2672–2680.

[2] L. Liu et al. "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* (2018).

[3] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[4] Ross Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[5] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[6] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[7] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[8] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv e-prints* (2018).

[9] A. Bochkovskiy, C. Y. Wang, and Hym Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: (2020).

[10] W. Liu et al. "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision*. 2016.

[11] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.

[12] X. Zhu et al. "Deformable DETR: Deformable Transformers for End-to-End Object Detection". In: (2020).

[13] Z. Yun, C. Zhi-Tong, and N. Tao. "Reverse modeling strategy of aero-engine blade based on design intent". In: *The International Journal of Advanced Manufacturing Technology* 81.9-12 (2015), pp. 1781–1796.

[14] Z. Zhao et al. "Measurement-based geometric reconstruction for milling turbine blade using free-form deformation". In: *Measurement* 101 (2017), pp. 19–27.

[15] Y. Feng, J. Ren, and Y. Liang. "Prediction and reconstruction of edge shape in adaptive machining of precision forged blade". In: *International Journal of Advanced Manufacturing Technology* 96.5-8 (2018), pp. 2355–2366.

[16] H. Yu, L. Xuegeng, and P. Liu. "Stream surface reconstruction of aero engine blade based on limited measured points". In: *Advances in Engineering Software* 131.MAY (2019), pp. 90–101.

[17] Y. Zhang, Z. Chen, and Z. Zhu. "Adaptive machining framework for the leading/trailing edge of near-net-shape integrated impeller". In: *International Journal of Advanced Manufacturing Technology* 107.9 (2020).

[18] D. Wu et al. "Research on adaptive CNC machining arithmetic and process for near-net-shaped jet engine blade". In: *Journal of Intelligent Manufacturing* 31.3 (2020), pp. 717–744.

[19] Bharat Singh and Larry S. Davis. "An Analysis of Scale Invariance in Object Detection - SNIP". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3578–3587. DOI: 10.1109/CVPR.2018.00377.

[20] Bharat Singh, Mahyar Najibi, and Larry S. Davis. "SNIPER: Efficient Multi-Scale Training". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 31 (NIPS 2018)*. Ed. by Bengio, S and Wallach, H and Larochelle, H and Grauman, K and CesaBianchi, N and Garnett, R. Vol. 31. Advances in Neural Information Processing Systems. 32nd Conference on Neural Information Processing Systems (NIPS), Montreal, CANADA, DEC 02-08, 2018. 2018.

[21] Wenhua Zhang et al. "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*. 2019, pp. 330–333. DOI: 10.1109/IGARSS.2019.8897842.

[22] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[23] Wenchao Liu, Long Ma, and He Chen. "Arbitrary-Oriented Ship Detection Framework in Optical Remote-Sensing Images". In: *IEEE Geoscience and Remote Sensing Letters* 15.6 (2018), pp. 937–941. DOI: 10.1109/LGRS.2018.2813094.

[24] H. Law and J. Deng. "CornerNet: Detecting Objects as Paired Keypoints". In: *International Journal of Computer Vision* 128.3 (2020), pp. 642–656.

[25] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. "Bottom-Up Object Detection by Grouping Extreme and Center Points". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 850–859. DOI: 10.1109/CVPR.2019.00094.

[26] X. Zhou, D. Wang, and P Krhenbühl. "Objects as Points". In: *arXiv preprint* (2019).

[27] X. Zhou, V. Koltun, and P Krhenbühl. "Probabilistic two-stage detection". In: *arXiv preprint* (2021).

[28] Ruchan Dong et al. "Sig-NMS-Based Faster R-CNN Combining Transfer Learning for Small Target Detection in VHR Optical Remote Sensing Imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.11 (2019), pp. 8534–8545. DOI: 10.1109/TGRS.2019.2921396.

[29] Zhaowei Cai and Nuno Vasconcelos. "Cascade R-CNN: Delving Into High Quality Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6154–6162. DOI: 10.1109/CVPR.2018.00644.

[30] Tom Young et al. "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]". In: *IEEE Computational Intelligence Magazine* 13.3 (2018), pp. 55–75. DOI: 10.1109/MCI.2018.2840738.

[31] Ashish Vaswani et al. "Attention Is All You Need". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30 (NIPS 2017)*. Ed. by Guyon, I and Luxburg, UV and Bengio, S and Wallach, H and Fergus, R and Vishwanathan, S and Garnett, R. Vol. 30. Advances in Neural Information Processing Systems. 31st Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, DEC 04-09, 2017. 2017.

[32] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 213–229. ISBN: 978-3-030-58452-8.

[33] Z. Sun et al. "Rethinking Transformer-based Set Prediction for Object Detection". In: *arXiv preprint arXiv:2011.10881v1* (2020).

[34] Jifeng Dai et al. "Deformable Convolutional Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 764–773. DOI: 10.1109/ICCV.2017.89.

[35] Rhr Hahnloser et al. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." In: *Nature* 405.6789 (2000), pp. 947–951.

[36] Ian J. Goodfellow et al. "Maxout Networks". In: *arXiv preprint arXiv:1302.4389v1* (2013).

[37] S. Elfwing, E. Uchibe, and K. Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning". In: *Neural Netw* (2018).

[38] Yinpeng Chen et al. "Dynamic ReLU". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 351–367. ISBN: 978-3-030-58529-7.

[39] N. Ma, Zhang X, and J. Sun. "Activate or Not: Learning Customized Activation". In: *arXiv:2009.04759v2* (2020).

[40] A. G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861v1* (2017).

[41] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[42] Andrew Howard et al. "Searching for MobileNetV3". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1314–1324. DOI: 10.1109/ICCV.2019.00140.

[43] Xiangyu Zhang et al. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6848–6856. DOI: 10.1109/CVPR.2018.00716.

[44] Ningning Ma et al. "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, 2018, pp. 122–138. ISBN: 978-3-030-01264-9.

[45] Kai Han et al. "GhostNet: More Features From Cheap Operations". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1577–1586. DOI: 10.1109/CVPR42600.2020.00165.

[46] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Springer International Publishing, 2014, pp. 740–755.

[47] Z. Qin et al. "FcaNet: Frequency Channel Attention Networks". In: *arXiv preprint arXiv:2012.11879* (2020).

[48] K. Chen et al. "MMDetection: Open MMLab Detection Toolbox and Benchmark". In: *arXiv preprint arXiv: 1906.07155* (2019).