Field rapid detection method of wind turbine blade fixing bolt defects based on FPGA

HOU Yupeng, ZHANG Lei*, WANG Yuanquan, ZHAO Xiaosong, FENG Guoce, and ZHANG Yirui
School of Artificial Intelligence, Hebei University of Technology, Tianjin 300130, China

(Received 23 March 2022; Revised 16 May 2022)

In order to solve the problem that blade fixing bolt cannot be detected quickly and conveniently in the field in actual production, this paper proposed a field rapid detection method of wind turbine blade fixing bolt defects based on field programmable gate array (FPGA), and Yolov4-tiny is selected as the basic algorithm. Nonetheless, the original Yolov4-tiny was not suitable for detecting small defects, so this paper improved the Yolov4-tiny to enhance the detection effect. Next, the convolutional operations in the algorithm were encapsulated into intellectual property (IP) cores by high-level synthesis (HLS) and Vivado, and parallel computation was realized using FPGA features. In the end, using Python to call the IP core and the FPGA hardware library, this paper achieved the purpose of rapid detection. Compared with traditional detection methods and other algorithms, the accuracy and speed of this method are significantly improved, which has a good application value.

Document code: A **Article ID:** 1673-1905(2022)09-0541-6
DOI https://doi.org/10.1007/s11801-022-2044-3

Fixing bolts have been widely used in wind turbine blades due to their useful properties, such as high strength, corrosion resistance, and stability[1]. In recent years, with the increasing number of wind turbines all over the world, the problem of wind turbine blades safety has become increasingly serious. However, many wind turbines are exposed to harsh environments all year round and are in a long-term vibration state, which leads to defects on the blade fixing bolts. At present, in order to ensure the safe operation of wind turbines, the relevant departments mainly use traditional methods such as ultrasonic flaw or computer vision to determine whether defects or not. However, the defects location of ultrasonic detection is inaccurate and requires extensive work experience, so the practicality is poor. Nowadays, the defects detection algorithm based on computer vision has become the focus of research[2], but with the continuous development of neural networks, its computation is increasing, and the requirements for the computational power of devices are getting higher and higher. The advanced RISC machine (ARM)[3] is difficult to cope with such a large amount of computational algorithm. The graphics processing unit (GPU) is competent, but it is not available for field detection. The field programmable gate array (FPGA) has a powerful computing power[4], which is very suitable for the computation of neural networks, and FPGA will become the backbone of artificial intelligence in the future[5]. In this paper, a field rapid detection method based on FPGA for fixing bolt of the wind turbine blade is proposed. This method is simpler than the ultrasonic method, more convenient than the GPU method, and faster than the ARM method. At the same time, this method also established a new work mode for workers, solved the problem that the blade fixing bolts in actual production cannot be detected quickly and conveniently in the field.

The Yolov4-tiny method is designed on the basis of the Yolov4 method, which is a simplified version of Yolov4. Yolov4 has about 60 million parameters, while Yolov4-tiny is reduced by nearly 90% and its inference speed is increased by 6—8 times[6]. Moreover, its weight file is only a few tens of MB, which greatly increases the feasibility of deploying object detection methods in embedded systems such as FPGA. Yolov4-tiny architecture can be broken down into three blocks. The first block is backbone network. Yolov4-tiny's backbone network Darknet53-tiny is mainly composed of the Darknet-Conv2D_BN_Leaky module and Resblock_body module stacked. The second block is the feature pyramid network (FPN), which is used to separate important features extracted from the backbone network. Yolov4-tiny head is the third block, which uses dense prediction for anchor-based detection that helps in dividing the image into multiple cells and inspecting each cell to find the probability of having an object using the post-processing techniques[7]. The network architecture of Yolov4-tiny is shown in Fig.1.

Due to the Yolov4-tiny detection network is relatively simple, although it has a fast detection speed, it is suitable for deployment of FPGA, its feature extraction

* E-mail: tjhouyupeng@163.com

process is simple, and the detection accuracy is low, especially in the indistinguishable small defects detection, the miss detection is more serious[8]. Therefore, it is difficult to meet the actual demand. In this paper, the spatial pyramid pooling (SPP) module and the path aggregation network (PAN) module are combined to improve detection capability for indistinguishable small defects. The improved Yolov4-tiny structure is shown in Fig.2.

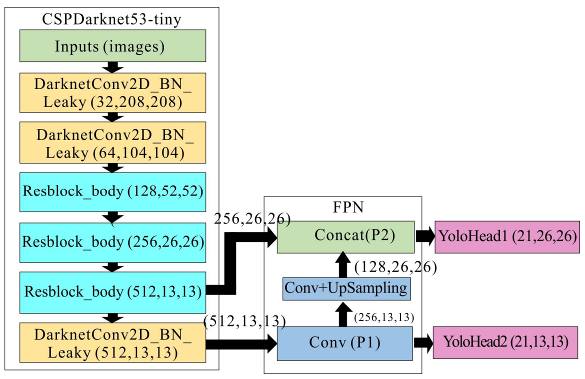
CSPDarknet53-tiny
Inputs (images)
DarknetConv2D_BN_ Leaky (32,208,208)
DarknetConv2D_BN_ Leaky (64,104,104)
Resblock_body (128,52,52)
Resblock_body (256,26,26)
Resblock_body (512,13,13)
DarknetConv2D_BN_ Leaky (512,13,13)
256,26,26)
(512,13,13)
FPN
Concat(P2)
(128,26,26)
Conv+UpSampling
(256,13,13)
Conv (P1)
YoloHead1 (21,26,26)
YoloHead2 (21,13,13)

Fig.1 Yolov4-tiny network structure

Defects are often not accurately detected because the size of the defects in the original image is not perfectly uniform. Therefore, this paper added the SPP module after the backbone network to increase the model's ability to achieve multiple feature map variations in detecting various defect sizes on all scales. The principle of the SSP module is to convert feature maps of arbitrary size into feature vectors of fixed size. The formulas are as follows

$$P_{\mathrm{h}}=\left\lfloor\frac{k_{\mathrm{h}}*n-h_{\mathrm{in}}+1}{2}\right\rfloor, \tag{1}$$

$$P_{\mathrm{w}}=\left\lfloor\frac{k_{\mathrm{w}}*n-w_{\mathrm{in}}+1}{2}\right\rfloor, \tag{2}$$

where k_{h} and k_{w} represent the length and width of the convolution kernel, n is the number of pooling, h_{in} and w_{in} represent the length and width of the output image, and P_{h} and P_{w} represent the numbers of padding in the length and width directions, respectively, to make the image output a uniform size after the SPP module. The processing steps of the SPP module are as follows. Firstly, the input feature layer undergoes three convolution processing (the convolution kernel size is 1×1, 3×3, and 1×1 respectively), and the number of channels is reduced from 512 to 256. Secondly, SPP uses three pooling layers (the kernel size is 3×3, 7×7, and 11×11, respectively) to perform maximum pooling processing, and the number of output channels of each pooling layer is 256. Thirdly, SPP stacks the input of the previous step and the output processed by the three pooling layers in the channel dimension, and the number of channels becomes 1 024. Finally, after three convolution processed, the number of channels output by the SPP module is restored from 1 024 to 256. While the SPP module performs multi-scale pooling and fusion on the input feature layer, it also greatly enhances the receptive field of the network and greatly improves the information extraction capability of the input feature layer[9].

Because CSPDarknet53-tiny of Yolov4-tiny has only two output feature layers converted to the predictive feature layer YoloHead, it is necessary to send the feature information to the FPN module for fusion to enhance the recognition ability of indistinguishable defects. However, the FPN structure of Yolov4-tiny is too simple, and the detection accuracy is not ideal. Based on the idea of feature fusion, this paper improved the network structure of Yolov4-tiny and used PAN to replace the FPN module. Compared with FPN, the strategy of repeated feature extraction and mutual fusion of PAN is more complicated, the PAN includes two paths of bottom-up and top-down feature fusion. The formula for calculating top-down path aggregation is shown as

$$M_i=\mathrm{conv}(Up_{\times 2}(P_{i+1})+M_{i+1}). \tag{3}$$

The bottom-up path aggregation formula is

$$M_i=\mathrm{conv}(Down_{\times 2}(P_{i-1})+N_{i-1}). \tag{4}$$

The resulting aggregated layer is

$$C_i=\begin{cases}M_{i+1}+P_i & i=2\\ M_i+N_i+P_i & 4\geq i\geq 3,\\ N_{i-1}+P_i & i\geq 5\end{cases} \tag{5}$$

where M_i is the top-down feature layer, N_i is the bottom-up feature layer, C_i is the fused feature layer, and P_i is the output of the backbone network.

After the introduction of the SPP module, the second output feature layer of the backbone network CSPDarknet53-tiny will pass through the SPP module firstly and then input the feature enhancement network PAN. Yolov4-tiny contains two prediction scales only, 13×13 and 26×26. Two feature maps are output from the deeper network, while the deeper network is easy to lose the shallow edge information. For PAN, it also has two input feature layers, F1 comes from the third Resblock_body module of the backbone network CSPDarknet53-tiny, and F2 comes from the SPP module. In the bottom-up fusion path, after the input F2 is subjected to 1×1 convolution processing and up sampling processing, it is stacked with the input F1 that has also undergone convolution processing in the channel dimension, and the number of channels in the feature layer increases from 128 to 256. The first output of PAN is obtained after the feature layer undergoes three convolutions (with the convolution kernel sizes of 1×1, 3×3, and 1×1, respectively). In the same way, in the top-down path, the output of the first path is stacked with the input F2 in the channel dimension after down sampling to complete the compression of height and width, and after three convolutions are completed, the first PAN obtains two outputs. The YoloHead is composed of a DarknetConv2D_BN_Leaky module with a convolution kernel size of 3×3 and an ordinary two-dimensional convolutional layer with a convolution kernel size of 1×1. The output is mainly used for the prior frame parameter adjustment. PAN uses the idea of bidirectional fusion to construct a bottom-up and top-down bidirectional channel to carry out a bidirectional fusion of information from diverse layers of the trunk network, so as to alleviate the loss of feature information caused by too many network layers. The features from different scales are fused and repeatedly enhanced, and the detailed information extracted by the backbone network can be fully utilized, which can greatly improve the detection accuracy of this algorithm.

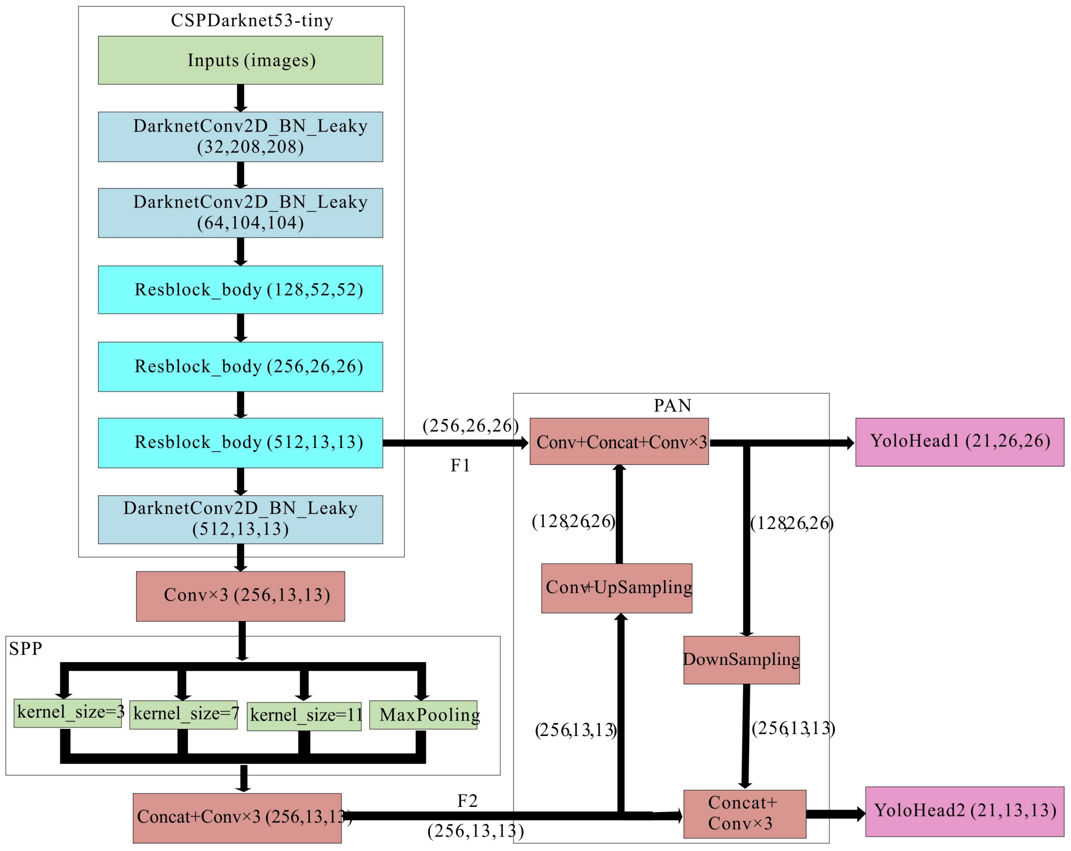

Fig.2 Improved Yolov4-tiny network structure

This paper used Python productivity for Zynq (PYNQ-Z2) embedded SOC developed by XILINX company. PYNQ is the first FPGA platform that supports Python. XILINX designed the Jupyter Notebooks environment for PYNQ[10]. PYNQ supports converting intellectual property (IP) cores developed by high-level synthesis (HLS) into overlay, and PYNQ provides Python library to call overlay to achieve the purpose of Python calling IP cores. PYNQ provides a complete underlying hardware library file, and developers can implement Yolo algorithm directly on Jupyter Notebooks without designing the underlying logic circuit. PYNQ consists of two parts, ARM and FPGA. The two parts are completely heterogeneous computational tasks. The PS where the ARM of the PYNQ-Z2 development platform is located is used for logic control to perform tasks with high flexibility and low computational load, including loading the Linux system, FPGA basic hardware library, Ipython kernel and corresponding application programming interface (API), and Jupyter Notebooks web server, etc. PYNQ uses the PL where the FPGA is located to perform the complex operations of the convolutional neural networks, and at the same time completes the analysis of the control commands and generate corresponding control signals to coordinate the operation of the internal modules. Its architecture is illustrated in Fig.3.

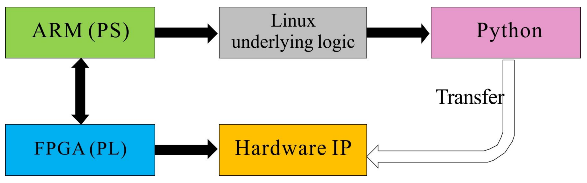

Fig.3 PYNQ FPGA architecture diagram

Although the improved algorithm can improve the detection effect, it cannot be directly deployed on the PYNQ FPGA. In order to make the algorithm get a better detection effect on FPGA, it is necessary to parallelize the algorithm and encapsulate the IP core.

Neural networks contain a large number of convolution operations, which are highly time-consuming. FPGA has excellent parallel processing capability. If the convolution operation is parallelized, the operation speed of the algorithm on the FPGA could be significantly improved. The HLS adopts the design idea of PIPELINE. In the process of convolution calculation, the input/output feature maps of different channels have no data dependencies in the process of being involved in the calculation, so parallelization is feasible[11]. Usually, the convolution for the loop is processed by folding. Folding means that the same circuit is employed in each loop. The circuit is time-multiplexed in the loop. If the loop is expanded, it is equivalent to the circuit being by copying several copies, the effect of parallel computing can be achieved. The convolution loop can be parallelized by using PIPELINE. This statement is equivalent to unrolling the loop, that is, the same circuit is copied several times to complete the parallelization of the algorithm.

After the convolution operation is parallelized, it needs to be packaged into an IP core before it can be called in a Python program. Firstly, this paper simulated and synthesized the designed parallelized program to generate a convolution operation IP core by HLS. Secondly, this paper used the Vivado Block Design tool to optimize and functionally simulate the overall circuit, and generated bit files and tcl files for the designed circuit. Finally, we imported the bit files, tcl files, and hwh files with the same file names into the folder of PYNQ-Z2 FPGA[12].

Due to the particularity of the actual situation, bolt detection usually needs to be carried out on-site, so the detection environment is more complicated. In this paper, we built a custom dataset using wind farm field defect data. While collecting the image dataset, we used industrial cameras to quickly sample defective bolts regularly, bright, dark, and disturbing conditions. Then, the image data were preprocessed by data processing techniques to obtain the RGB image dataset in the format of VOC[13], with an image size of 416×416 and a number of 1 000 images. The images in the dataset were randomly selected according to the algorithm rules and assigned to the training set, test set, and validation set in the ratio of 7: 2: 1. The defect types were bolt cracks that were difficult to be distinguished directly by professionals, and the defects dataset is shown in Fig.4.

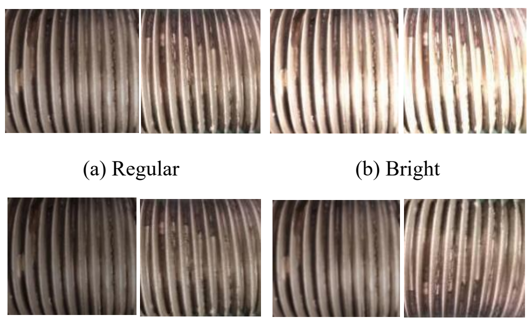

Fig.4 Dataset examples

The original Yolov4-tiny, Yolox-tiny, and the improved Yolov4-tiny algorithm are used for deep learning training in the PC (GTX 1060 GPU) environment with the Pytorch deep learning framework, and the number of iterations is 300. After the training is completed, according to the training information in the results, a visual drawing is performed to obtain the comparison chart of the loss value (Loss) and the mean average precision (mAP_0.5), as shown in Fig.5.

In Fig.5, the green curve is the Loss and mAP values of the original Yolov4-tiny algorithm, the blue curve is the Loss and mAP values when only the SPP module is added, the yellow curve is the Loss and mAP values when only the PAN module is replaced, the black curve is the Loss and mAP of Yolox-tiny, and the red curve is the final improved Loss and mAP values. In this figure, it can be seen intuitively that the Loss and mAP values of the improved algorithm have improved to a certain extent[14]. With the increase of the number of iterations, the loss value is constantly decreasing. The performance shown by Loss and mAP in the training session shows that Yolov4-tiny-SPP+PAN achieves the best performance, its mAP is up to 97.2%, and the Loss is less than 0.7. By combining the mAP and Loss metrics, it can be inferred that the improved Yolov4-tiny network model is ideal for the training session.

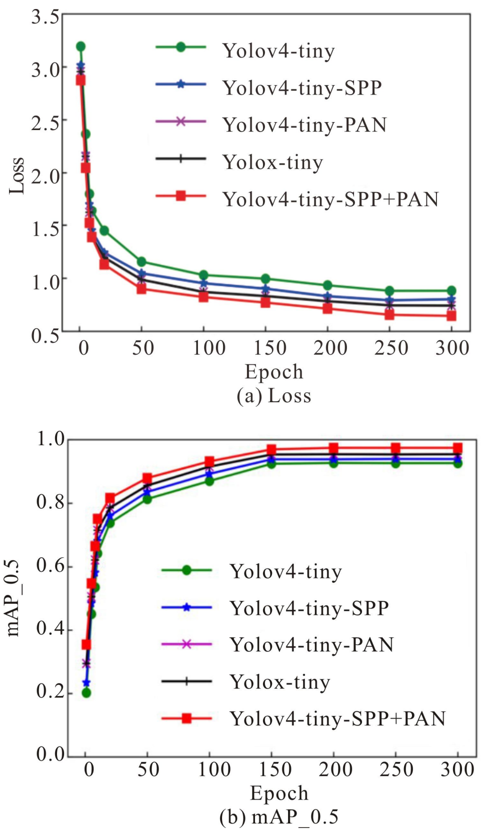

Fig.5 Model performance evaluation results

To evaluate the improved Yolov4-tiny performance, different structural improvement experiments have been conducted and the impact of each improved part of the algorithm on the model training and detection accuracy has been listed in this paper, as shown in Tab.1. It's obvious that the SPP+PAN module significantly improved the feature extraction for Yolov4-tiny. What's more, this paper also used another two indicators to calculate the detection accuracy and detection speed of three platforms respectively. The evaluation indicators are the mAP value and FPS value. The formulas are as follows

$$\mathrm{mAP} = \frac{1}{n}\sum_{i=1}^{n} AP_i, \tag{6}$$

$$\mathrm{FPS} = \frac{FigureNumber}{TotalTime}. \tag{7}$$

In the above formula, mAP refers to the mean accuracy obtained by dividing the area of the RP curve by the number of defect categories, and FPS refers to the total number of inspection pictures divided by the detection time[15]. The corresponding data are shown in Tab.2.

After the device is connected correctly, the specific detection steps are as follows. First, the location of the device should be determined. We placed the device on a bench or flat ground in the field wind turbine cabin, connect the power supply and display. Then, we run the top-level Python program in Jupyter Notebooks, and placed the bolt to be tested about 5 cm away from the FPGA hardware platform, then the FPGA will detect it. It is necessary to manually rotate and adjust the position of the bolt to check whether all sides of the bolt are normal. Finally, the detection results will be displayed on Jupyter Notebooks. The detection results are shown in Fig.6.

The results achieved by the improved algorithm as in Tab.2 shows that performance is better than the original algorithm, where the accuracy reaches 97.2% compared to the original algorithm 90.3%. Although the detection effect of the same algorithm is reduced by about 2% on the PYNQ FPGA platform compared to other platforms, the FPS reaches 46.6 compared to the GPU 32.0 and ARM 23.2. Therefore, the comprehensive performance of the improved Yolov4-tiny on FPGA is superior.

Tab.1 Comparison of Loss and mAP values of different structures and algorithms

Indicators	Yolov4-tiny	Yolov4-tiny-SPP	Yolov4-tiny-PAN	Yolox-tiny	Yolov4-tiny-SPP+PAN
Loss	0.84	0.76	0.70	0.68	0.61
Recall (%)	92.3	93.1	95.1	94.5	96.9
Precision (%)	92.1	92.7	95.8	95.3	97.4
mAP_0.5 (%)	90.3	92.5	95.6	96.1	97.2

Tab.2 Comparison of algorithm improvement effects on different platforms

Platform types	Original algorithm average detection accuracy (%)	Improved algorithm average detection accuracy (%)	Original algorithm average detection speed (fps)	Improved algorithm average detection speed (fps)
PYNQ FPGA	89.6	95.3	47.8	46.6
GTX1060 PC	90.3	97.2	34.5	32.0
Cortex-A9 ARM	90.1	96.9	25.7	23.2

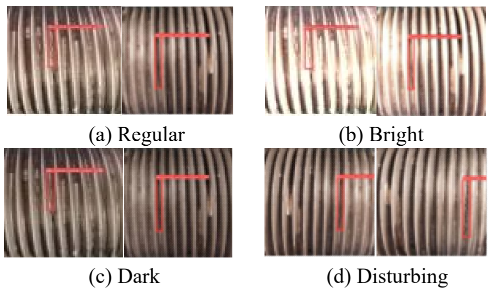

Fig.6 Example of detection results

Tab.3 Logical resource utilization

Resource	Utilization	Available
LUT	10 274	53 200
LUTRAM	614	17 400
FF	6 889	106 400
BRAM	79	140
DSP	185	220

In addition, this paper explored the resource utilization of PYNQ FPGA, based on comprehensive reports provided by the Vivado tool[16]. It can be seen from Tab.3 that there are many remaining resources available on the PYNQ FPGA, so the resource utilization of the improved algorithm in this paper is reasonable.

As the traditional bolt defects detection methods cannot achieve field rapid detection in wind farms, this paper proposed a field rapid detection method for wind turbine blade fixing bolt defects based on FPGA. This paper made structural improvements to Yolov4-tiny, that is, adding the SPP module and replacing FPN with PAN to improve the detection ability of Yolov4-tiny. Then the improved Yolov4-tiny is ported to FPGA by HLS, Vivado, and FPGA technology. Experimental results showed that the field rapid detection method for wind turbine blade fixing bolts based on FPGA is better and more convenient than the traditional methods, and the model of improved Yolov4-tiny is better than the model of original Yolov4-tiny and Yolox-tiny, so this method has strong field adaptability and wide application value. However, this paper still has many shortcomings, for example, although the improved Yolov4-tiny can enhance the detection accuracy, it will increase the compu tation, which may deviate from the original purpose of Yolov4-tiny in order to reduce the number of parameters. In addition, this paper has less optimization effort for the algorithm on the FPGA.

Statements and Declarations

The authors declare that there are no conflicts of interest related to this article.

References

[1] YU G A, QIN Z W, RONG X M, et al. Research on the influence of defects on the performance of bolted connections of wind turbine blades[J]. Acta Energiae Solaris Sinica, 2019, 40(11)：3244-3249. (in Chinese)

[2] TAO X, HOU W, XU D, et al. A review of surface defect detection methods based on deep learning[J]. Acta Automatica Sinica, 2021, 47(05)：877-879. (in Chinese)

[3] YU W Y, ZHANG Y, YAO H M, et al. Visual inspection of surface defects based on lightweight reconstruction network[J]. Acta Automatica Sinica, 2020, 41(16)：1-12. (in Chinese)

[4] CHEN C, CHAI Z L, XIA J. Design and implementation of YOLOv2 accelerator based on Zynq 7000 FPGA heterogeneous platform[J]. Journal of frontiers of computer science & technology, 2019, 13(10)：1677-1693. (in Chinese)

[5] ADIONO T, PUTRA A, SUTISNA A, et al. Low latency YOLOv3-tiny accelerator for low-cost FPGA using general matrix multiplication principle[J]. IEEE access, 2021, 9(08)：141890-141913.

[6] ZHU J, WANG J L, WANG B. Lightweight mask detection algorithm based on improved YOLOv4-tiny[J]. Chinese journal of liquid crystals and displays, 2021, 36(11)：1525-1534. (in Chinese)

[7] LI F D, GAO D Y, YANG Y Q. Small target deep convolution recognition algorithm based on improved YOLOv4[J]. International journal of machine learning and cybernetics, 2022, 3(12)：982-990.

[8] ADIBHATLA A, CHIH H, HSU C, et al. Defect detection in printed circuit boards using you-only-look-once convolutional neural networks[J]. Electronics, 2020, 9(09)：1547-1563.

[9] ADDIE I B, TOFAEL A. Real time pear fruit detection and counting using YOLOv4 models and deep SORT[J]. Sensors, 2021, 21(14)：4803-4811.

[10] MÁNDI Á, MÁTÉ J, RÓZSA D, et al. Hardware accelerated image processing on FPGA based PYNQ-Z2 board[J]. Carpathian journal of electronic and computer engineering, 2021, 14(01)：20-23.

[11] BJERGE K, SCHOUGAARD J H, LARSEN D E. A scalable and efficient convolutional neural network accelerator using HLS for a system-on-chip design[J]. Microprocessors and microsystems, 2021, 87(03)：198-206.

[12] HUYNH T V. FPGA-based acceleration for convolutional neural networks on PYNQ-Z2[J]. International journal of computing and digital systems, 2022, 11(01)：441-450.

[13] MENG T, TAO Y, CHEN Z Q. Depth evaluation for metal surface defects by eddy current testing using deep residual convolutional neural networks[J]. IEEE transactions on instrumentation and measurement, 2021, 70(02)：1862-1871.

[14] BOUGUEZZI S, BEN F H, BELABED T, et al. An efficient FPGA-based convolutional neural network for classification：Ad-MobileNet[J]. Electronics, 2021, 10(18)：2272-2283.

[15] REN Z H, FANG F Z, YAN N, et al. State of the art in defect detection based on machine vision[J]. International journal of precision engineering and manufacturing-green technology, 2021, 2(06)：1-31.

[16] LIU J, GE Y F. Reconfigurable convolutional neural network accelerator based on ZYNQ[J]. Chinese journal of electronics, 2021, 49(04)：729-735.