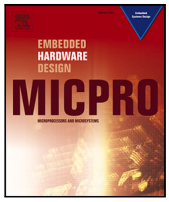
Microprocessors and Microsystems 100 (2023) 104863

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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



Real-time detection of track fasteners based on object detection and FPGA

Tian Xiao [a], Tianhua Xu [a,*], Guang Wang [b]

[a] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China
[b] Department of Computer Science, Florida State University, Tallahassee, Florida, USA

ARTICLE INFO

Keywords:
Fastener detection
Real-time
Convolutional neural network
YOLOv2 detection network
FPGA

ABSTRACT

An accurate and fast fasteners detection is of great significance for improving the inspection efficiency of railway tracks. However, the task is also challenging due to the limited memory and processing capacity of the embedded maintenance systems. To address these challenges, in this paper, we presents a solution that can be deployed on the high-speed operation vehicles, including You Only Look Once version 2 (YOLOv2) network and Field Programmable Gate Array (FPGA) HW-based acceleration strategy. The algorithm proposed in this paper is implemented completely on FPGA, and the hardware acceleration strategy such as loop pipelining, array partition and ping-pong buffer optimization is applied to get the highest performance. The hardware implementation results show that the improved YOLOv2 detection network and fastener detection system are superior to the existing embedded maintenance system in processing time and hardware resources. The proposed systems can detect a track image in 41 ms, which is much faster than the existing embedded maintenance system.

1. Introduction

In recent years, the railway system in China has developed rapidly [1]. Heavy traffic and high speed pose challenges to the stability and safety of the railway system. Track fastener is an intermediate part to ensure that rail and sleeper do not deviate or separate. Tracks and other infrastructure have been exposed to the natural environment and have been affected by extreme weather all year round. Therefore, changes in track shape and damage to connecting devices have always affected the safety of railway line operations [2]. In addition, the contact friction and vibration between the train wheel and the track are also easy to cause rail fracture or fasteners missing and other defects [3]. Once the fastener is defective or missing, the rail may move horizontally or longitudinally relative to the sleeper, leading to train derailment and other major accidents [4]. Therefore, it is very important to detect the position of rail fasteners and obtain their health status in real time.

At present, the ways to realize track fastener detection are mainly divided into two categories: (i) manual detection and (ii) track inspection vehicle detection. However, both of the two methods have real-world problems. For example, the manual inspection is inefficient and costly, which is hard to meet the requirement of rapid development of railway system [5]. For the track inspection vehicle, its main disadvantages are the long inspection cycle due to a limitation of inspection vehicle, and expensive [6]. In order to solve these problems, the study for a railway track fastener detection equipment that has a small size, low power consumption and fast detection speed to improve the safety and stability of railway systems, is currently proceeding. The detection equipment with above feature can effectively improve the inspection efficiency and take up little space.

At an early stage, traditional image process methods were used for railway fasteners detection. De Ruvo et al. [7] propose a prototype architecture based on FPGA and implement a prediction algorithm for automatic detection of the existence of fastening bolts. Singh et al. [8] used edge density for detecting missing fasteners from videos, which can replace manual visual inspection to improve inspection efficiency. Yang et al. [9] extracted the direction field as the feature descriptor of the fastener, and detected the missing fastener by weighted template matching. Feng et al. [10] proposed an automatic visual inspection system for detecting fasteners by using a probabilistic topic model based on the Haar-like feature. Although the above methods improve the detection accuracy to a certain extent, the improvement is limited

With the development of deep learning technology, deep learning is widely used to replace shallow learning network to obtain key information. Dong et al. [11] introduced a convolutional neural network (CNN) model for classifying fasteners. Liu et al. [12] adopted a similarity-based deep learning network to classify the categories of fasteners and achieved good classification results. Although the above methods can achieve good detection accuracy, the detection speed still needs to be improved.

In order to improve the detection speed of track fasteners, many lightweight networks have been applied to track detection tasks. Chen

* Corresponding author.
E-mail address: thxu@bjtu.edu.cn (T. Xu).

https://doi.org/10.1016/j.micpro.2023.104863
Received 3 September 2021; Received in revised form 27 August 2022; Accepted 10 May 2023
Available online 19 May 2023

et al. [13] proposed a novel network architecture deep convolutional neural network (DCNN) to detect defects in fastener images, which cascades three DCNN-based detection stages to localize the cantilever joints and their fasteners. Li et al. [14] proposed a YOLOv3-Lite detection method, which combines deep separable convolutions and feature pyramids to improve the precision of defect detection. Xiao et al. [15] introduced a residual network structure into the YOLOv3-Tiny network, which improved the detection accuracy of the target. Although these methods perform well in their assigned tasks and can further simplify the detection process, they are not suitable for embedded systems. Therefore, the above methods cannot be applied to the real-world track inspection system directly.

In order to solve these problems, we propose to deploy the YOLOv2 [16] object detection network in the embedded system to realize automatic detection of track fasteners. The embedded system has the characteristics of small size and easy installation and deployment. YOLOv2 object detection network can ensure the detection accuracy, and it is accelerated by the embedded system to achieve real-time detection. FPGA has obvious advantages in data parallelism/ pipeline parallelism to reach microsecond latency, cost efficiency, lower power consumption and smaller size. Therefore, FPGA-based YOLOv2 network is employed in the railway track fastener detection equipment to meet the real-time application, limited space and power consumption. The corresponding C/C++ code is written by using Vivado High Level Synthesis (HLS), and the algorithm is implemented on hardware. The comparison between HardWare-based (HW-based) approach and HardWare/SoftWare Co-Design (HW/SW Co-Design) approach shows that the performance of HW-based solution is promising. The main objective of this paper is to present an innovative way to implement fastener detection algorithms on hardware to accelerate the execution time while minimizing power consumption. The main contributions are as follows:

- The proposed approach introduces a way to integrate the deep convolutional neural network into heterogeneous reconfigurable hardware.
- A optimizing acceleration strategy is proposed to reduce latency and resource consumption.
- More importantly, we deployed the proposed model in Zynq embedded platform and achieved real-time track fastener detection.

The rest of the paper is organized as follows: Section 2 describes our proposed system. The Section 3 introduces the YOLOv2 object detection network. The implementation of the YOLOv2 object detection network on the Zynq platform are presented in Section 4. A brief description of the Zynq platform is given as well as details about the implementation and design flow using HLS and Vivado suite. In Section 5, the experimental verification and the results comparison are given. Finally, Section 6 concludes the paper and highlights some perspectives of future work.

2. System overview

2.1. Zynq SoC platform

The Zynq System on Chip (SoC) platform is chosen for the hardware acceleration of the fastener detection system due to its flexibility and suitability for a HW/SW co-design approach. The Xilinx Zynq-7000 all programmable SoC combines a traditional FPGA based on Xilinx 7-series forming the Programmable Logic (PL) with a dual core ARM Cortex-A9 processor forming the Processing System (PS). The PL is based on Artix-7 or Kintex-7 with different variants. Details about the Zynq-7000 all programmable SoC can be found in [17]. The combination of the PS and the PL inside the same chip makes platforms based on the Zynq SoC suitable for HW/SW co-design approach. Especially when associated with Xilinx Vivado High Level Synthesis (HLS) tools, Vivado IP integrator, and Software Development Kit (SDK), it will allow high-level abstractions with advantages in performance, cost, and power compared to traditional FPGA or processor implementations. The starting point of the hardware/software co-design on Zynq SoC is the system design, which sets all the required requirements and specifications for the top-level system and various subsystems.

2.2. Track inspection system

A track inspection system typically consists of four high-definition cameralink industrial cameras and two laser lights in order to fully obtain the spatial information on both sides of the railway track [18]. Fig. 1 shows the relative spatial position and exposure range of each industrial camera. The Camera Ci (i = 1,3) is used to capture images of steel rails, which rely on image processing, feature matching and visual computing to find track image measurement points on the rail track. The Camera Ci (i = 2,4) is positioned directly above the track to capture the image of the track fastener. In our design, an image of the fastener is captured by two industrial cameras and sent to the designed track fastener detection system, where the object detection network method is used to detect the fastener.

3. YOLOv2 fasteners detection network

3.1. Network architecture

Compared with YOLO, YOLOv2 has made a lot of improvements, which also makes the map of YOLOv2 have been greatly improved, and the speed of YOLOv2 as a one-stage method is still very fast, which can be better applied to track fasteners detection. YOLOv2 uses a new base model (feature extractor) called Darknet-19, with 19 convolution layers and 5 maxpooling layers, as shown in Fig. 2.

Darknet-19 uses global avgpooling for prediction and uses $1 * 1$ convolution between $3 * 3$ convolution to compress the feature graph channels to reduce the computation and parameters of the model. Darknet-19 also uses the batch normalization layer behind each convolution layer to speed up the convergence and reduce the over-fitting of the model. YOLOv2 works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell. According to the ground truth and loss function, the position offset and size adjustment of the predicted bounding box are realized through training.

3.2. Loss function

The loss function of YOLOv2 is shown as below:

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{w*h}\sum_{j=0}^{n}1_{ij}^{obj}(2-w_i*h_i)\\
&\times[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2+(w_i-\hat{w}_i)^2+(h_i-\hat{h}_i)^2]\\
&+\sum_{i=0}^{w*h}\sum_{j=0}^{n}1_{ij}^{obj}(C_i-\hat{C}_i)^2+\lambda_{noobj}\sum_{i=0}^{w*h}\sum_{j=0}^{n}1_{ij}^{noobj}(C_i-\hat{C}_i)^2\\
&+\sum_{i=0}^{w*h}\sum_{j=0}^{n}1_{ij}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2
\end{aligned}\tag{1}$$

In the fastener detection model, the penalty coefficient λ_{coord} and λ_{noobj} of the coordinate error are set to 5 and 0.5, respectively. x_i, y_i, w_i, h_i and $\hat{x}_i$, $\hat{y}_i$, $\hat{w}_i$, $\hat{h}_i$ are the center coordinates and size of the real bounding box and the predicted bounding box, respectively. 1_{ij}^{obj} denotes whether the center of the detection fastener is contained in the ith grid where the jth candidate bounding box is located. C_i and $\hat{C}_i$ denotes the confidence value of cell i in the real bounding box and predicted bounding box, respectively. $p_i(c)$ and $\hat{p}i(c)$ denotes the conditional probability that grid i in the real bounding box and the predicted bounding box contains the c-type fasteners.

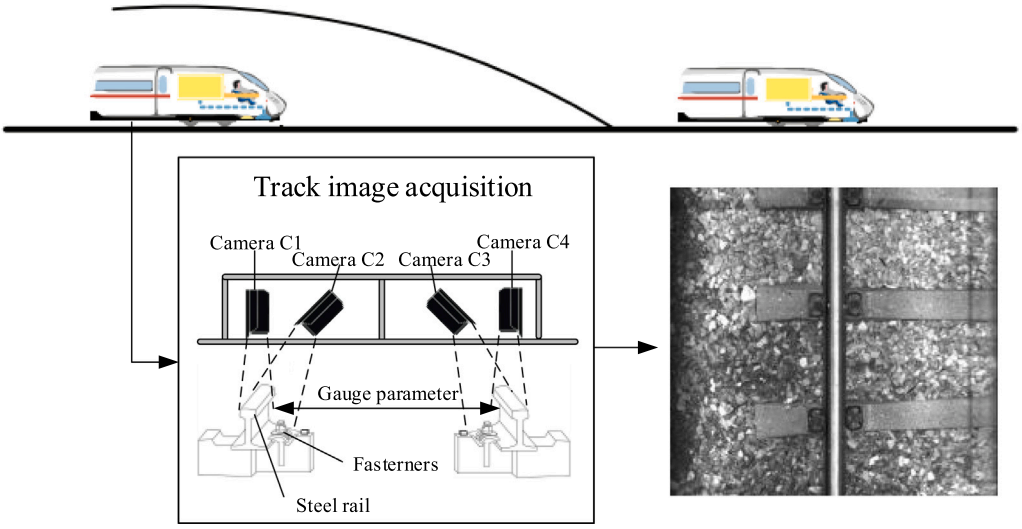


Fig. 1. Railway track inspection principle.

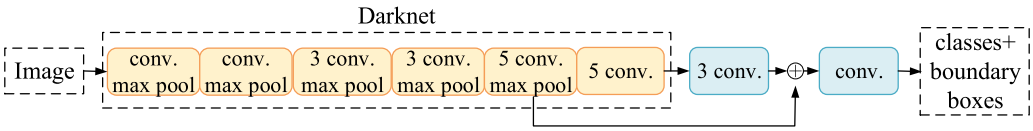


Fig. 2. YOLOv2 object detection model structure.

4. System implementation

The HW/SW Co-Design approach has data exchange between ARM processor and FPGA [19]. The time cost of data transfer will increase the time of fastener detection. Thus, our fastener detection algorithm using only an FPGA avoids the data exchange between FPGA and ARM processor and accordingly leads to higher system performance and faster detection speed. FPGA can provide high bandwidth for data and reduce the delay at the same time. There are three ways to design hardware accelerated IP core. One way is to use Vivado HLS [20], use C, C++ or system C to create and simulate IPs, and then export it. Another way is to use Xilinx system generator. Another way is to use hardware description language (HDL), such as VHDL or Verilog. In addition, using Xilinx system generator can also generate the hardware accelerated IP core. In all cases, the IPs are exported and added to the Vivado IP Catalog. Then use IP integrator to create a hardware system in Vivado, all modules are connected to each other, including Zynq SOC and the previously created IP.

4.1. Vivado HLS

Hardware accelerated IP core development process is based on C language, which can save users a lot of time. As shown in Fig. 3, its important processes include: C development, C simulation, C synthesis and RTL synthesis. Details about Vivado HLS can be found in [21].

The hardware block design when using Vivado IP Integrator is shown in Fig. 4. The overall design uses four IPs, including ZYNQ7 Processing System, Predict, AXI Interconnect, Processor System reset. ZYNQ7 Processing System is a fixed PS on the Zynq chip, used to specify anything related to PS. Predict refers to the IP designed, simulated and exported in HLS. AXI Interconnect is used to interconnect an AXI memory mapped master device. Processor System reset provides a customized resets for the entire system including the PS, the AXI interconnect core and the Predict core from HLS. Details about Vivado IP Integrator can be found in [22].

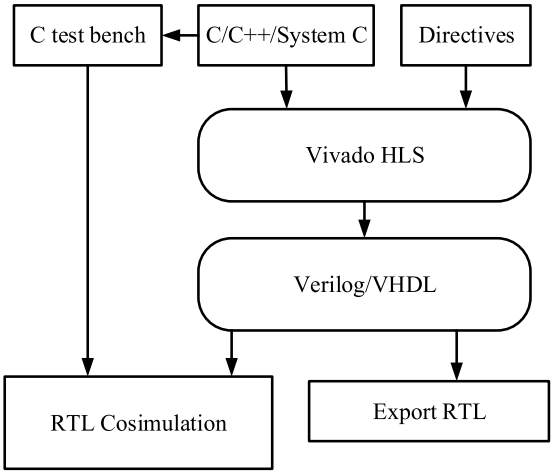


Fig. 3. Schematic diagram of Vivado HLS.

4.2. Inner layer parallel implementation

To implement a layer of the deep learning algorithm in FPGA, this layer needs to be reused during calculation [23]. After each layer is calculated, the data is cached in DDR. When the next layer is calculated, the data is read into the operation unit. In this process, the ARM of FPGA is needed to configure the IP core, including the number of input and output channels, the size of convolution core and so on. After configuration, IP and can carry out corresponding layer operation. The architecture of convolutional IP core is as shown in Fig. 5.

This system includes DDR, ARM processor, controller, arithmetic unit and various buffers. The input image is first loaded into the register, and then convolution operation is performed through the operation

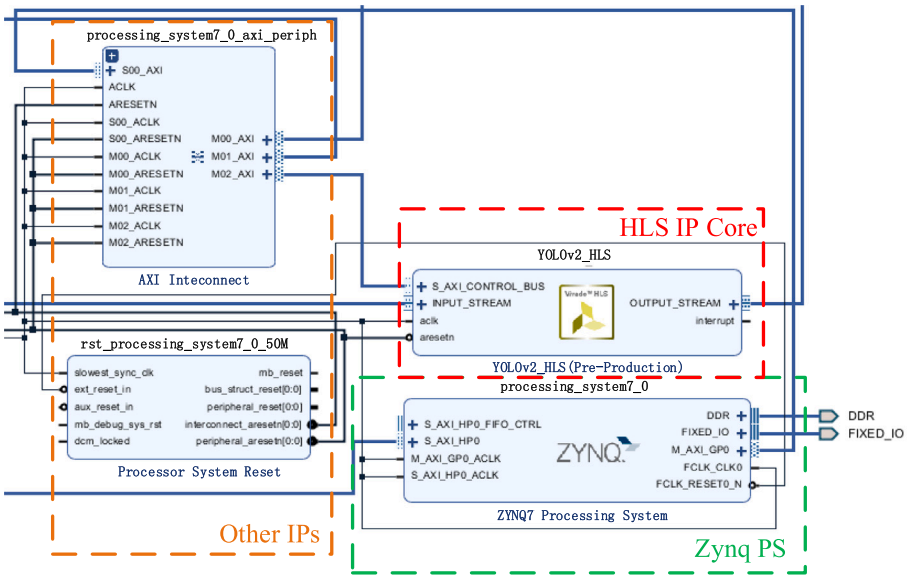


Fig. 4. Block design implementation on the chip.

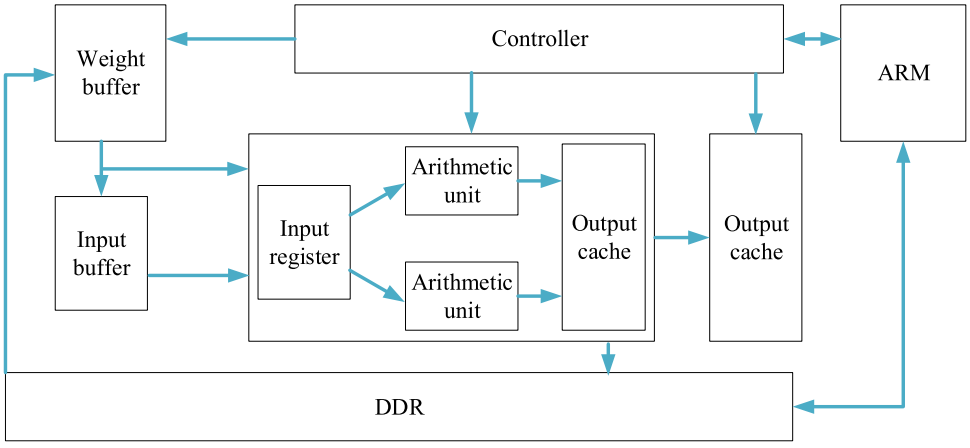


Fig. 5. Convolutional IP core architecture.

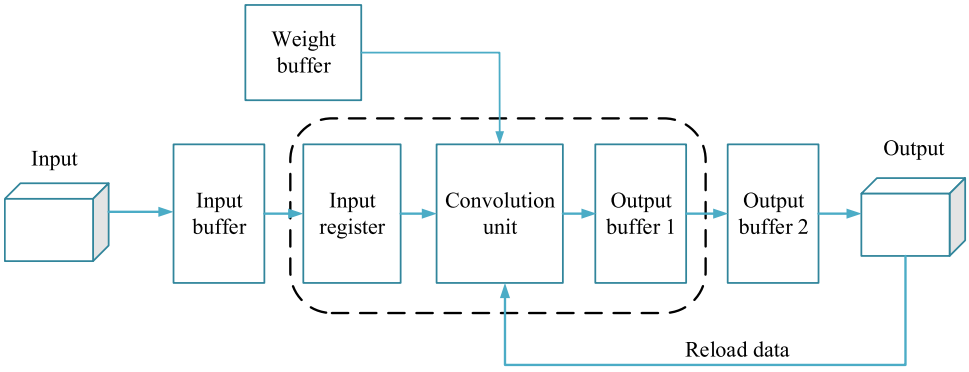
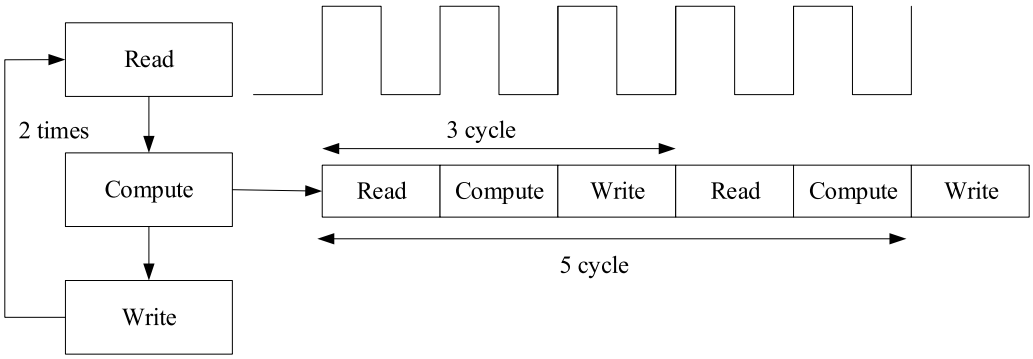


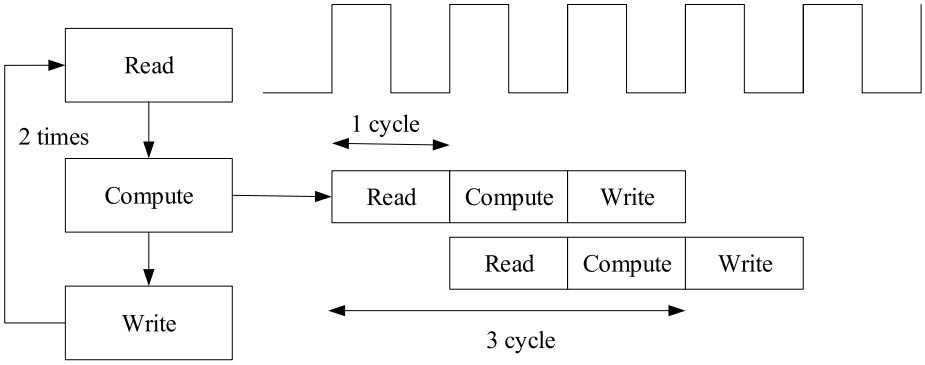
Fig. 6. Single layer optimization of the detection network.

unit. Convolution operation is performed through multiple operation units to ensure the speed of operation. In convolution operation, the data in the first level output cache will be output to the second level output cache. After the operation of the current layer is completed, the operation structure will become the operation output of the next level. In this way, each layer of the network can be accelerated.

As shown in Fig. 6, it can realize the parallel optimization of the network in the single layer internal operation, and it does not need to expand the whole network, so it can reduce the resource occupation rate of the network, and then reduce the power consumption. Furthermore, if we want to accelerate the single layer network, we need to implement pipelining in the layer.



(a)Without loop pipelining



(b)After loop pipelining

Fig. 7. Comparison before and after loop pipelining.

4.3. Cycle optimization

HLS has many optimization instructions for loops. Here we only use loop pipelining and loop unrolling. The function of loop pipelining is to pipelined parallel processing of loops [24]. In this way, the execution time of two loops can be overlapped, so that the next loop can be executed at the same time in the process of this loop.

As shown in Fig. 7, it is clear that if two iterations are taken as an example, the operation without loop pipelining needs 5 clock cycles to complete, while the operation after loop pipelining only needs 3 clock cycles. After loop pipelining, the initiation interval has been reduced from 3 clock cycles to 1 clock cycle.

Before loop optimization, the operation of the loop follows the default settings. But after the loop unrolling, the circuit of the loop will be set to N, which is usually specified by the instruction factor in HLS. For example, if the factor is 2 and the number of iterations is 8, the iterations will be divided into four times, and the inner times will be implemented in two loops at a time. Loop unrolling allows loops to be executed in parallel having dedicated hardware resources for each loop [25].

FPGA can realize parallel processing in the operation process. Therefore, in the convolution calculation process of the convolutional neural network, this paper parallelizes the process of the convolution operation. Compared with other dimensions, the input channel and output channel are easier to parallelize. As shown in Fig. 8, this process can be implemented with a PIPELINE statement.

As shown in Fig. 8, the convolution operation part uses #pragma HLS UNROLL to expand 27 multiplication and addition operations, and then execute them in parallel. The optimization target of this IP core is #pragma HLS PIPELINE II = 1, and the smaller the value of II, the faster the processing speed. Since the #pragma HLS UNROLL is located within the #pragma HLS PIPELINE II = 1, it will expand all sub-loops in the loop, so the #pragma HLS UNROLL instruction can be omitted.

```
for(int height=0;height<416+1;height++){
    for(int width=0;width<416+1;width++){
        #pragma HLS PIPELINE II=1
        for(int ch=0;ch<3;ch++){
            #pragma HLS UNROLL
            for(int i=0;i<9;i++){
                tmp_conv+=window[ch][i]*buffer_weights[ch][i];
            }
        }
    }
}
```

Fig. 8. Pseudo codes for loop optimization.

4.4. Array partition

Array partition is one of the key technologies to improve the bandwidth of memory. Because the array is implemented as memory, and it has only two data ports at most, which limits the read/write (or load/store) throughput in the pipeline [26]. In order to effectively improve throughput, the original array needs to be split into multiple smaller arrays that use multiple memory elements. In this way, each element of the original array is assigned to each new array. These arrays can run in parallel within the pipeline, which can significantly improve the throughput of the pipeline.

```
    data_t In[Tin][S*Tr+K-S][S*Tc+K-S];
#pragma HLS ARRAY_PARTITION variable=In complete dim=1
    data_t W[Tout][Tin][K][K];
#pragma HLS ARRAY_PARTITION variable=W complete dim=1
#pragma HLS ARRAY_PARTITION variable=W complete dim=2
    static data_t Out[Tout][Tr][Tc];
#pragma HLS ARRAY_PARTITION variable=Out complete dim=1
```

Fig. 9. Pseudo codes for array partition.

As shown in Fig. 9, this paper optimizes the array and expands the array in the specified dimension to provide multi-port read and write capabilities. In Fig. 9, $T_{in} = 8$, $T_{out} = 32$, $T_r = 13$, $T_c = 13$, $K = 3$ and $S = 1$. After the loop pipeline and array partition processing, this paper realizes the parallelization of the convolution operation process.

4.5. Ping-pong buffer

The Ping-pong buffer is designed to achieve seamless data buffering, data processing and save buffer space. The timing diagram of ping pong buffer is shown in Fig. 10. For the data read process, it reading the input feature map pixel/weight data from the off-chip DRAM to the ping and pong buffer alternately. The delay of reading data into ping/pong buffer overlaps with the computing process of pong/ping data, reducing the data transmission delay and improve the reuse of buffer space. It overlaps the delay L_{Load} of reading data from DRAM, the delay $L_{Compute}$ of data processing on chip and the delay L_{Store} of writing the processed data back to DRAM to reduce the total delay [27,28]. The total delay L of convolution layer and pooling layer without ping-pong buffer is shown in Eqs. (2) and (3). After ping-pong buffering, the total delay L of convolution layer and pooling layer is shown in Eqs. (4) and (5).

$$L = [\frac{R}{T_r}] \times [\frac{C}{T_c}] \times [\frac{M}{T_m}] \times (L_{Inner} + L_{Store}) \tag{2}$$

$$L = [\frac{R}{T_r}] \times [\frac{C}{T_c}] \times [\frac{M}{T_m}] \times (L_{Load} + L_{Compute} + L_{Store}) \tag{3}$$

$$L = [\frac{R}{T_r}] \times [\frac{C}{T_c}] \times \{([\frac{M}{T_m}] - 1) \times \max(L_{Inner}, L_{Store}) + L_{Inner} + L_{Store}\} \tag{4}$$

$$L = [\frac{R}{T_r}] \times [\frac{C}{T_c}] \times \{[\frac{M}{T_m}] \times \max(L_{Load}, L_{Compute}, L_{Store}) + L_{Load} + L_{Store}\} \tag{5}$$

$$L = ([\frac{N}{T_n}] - 1) \times \max(L_{Load}, L_{Compute}) + L_{Load} + L_{Compute} \tag{6}$$

where R, C, M, and N respectively represent the height of the output feature image, the width of the output feature image, the number of output feature image, and the number of input feature image. T_r and T_c are the width and height of the pixel block of the output feature map, respectively. T_m is the output parallelism, and T_n is the input parallelism. After ping-pong buffer, the three kinds of delay are overlapped, which greatly reduces the total delay and saves the buffer space.

4.6. Multi-channel data transmission

Considering that the actual bandwidth of DRAM is greater than the bus bandwidth of a single interface, multi-channel data transmission can be used to reduce transmission delay [29]. As shown in Fig. 11.

As shown in Fig. 11(a), the data distribution module becomes N sub-input modules, and each sub-input module reads the pixel blocks of Tn/N input feature maps from DRAM to the on-chip cache. In Fig. 11(b), the data collection module becomes M sub-output modules, and each sub-output module writes the pixel blocks of the Tm/M output feature maps from the on-chip buffer back to the off-chip. There is no dependency between the input and output sub-modules, and the data transmission is completed concurrently, which can greatly reduce the transmission delay.

Table 1
The size and proportion of different fasteners in the dataset.

Dataset	Size	Images	Proportion
Training	4096 * 4096	3252	60%
Verification	4096 * 4096	1086	20%
Test	4096 * 4096	1086	20%

Table 2
The detection performance of different object detection networks.

Model	mAP	FPS
Fast R-CNN	90.1%	5.1
Faster R-CNN	93.8%	10.4
Mask R-CNN	95.2%	5.7
YOLO	83.5%	45.1
SSD	94.2%	46.1
YOLOv2	99.3%	57.3

5. Experimental verification

5.1. Network training and experiment setting

The evaluation dataset was collected from the Ring Test Field of National Railway Test Center in Beijing, China. As shown in Table 1, the dataset is divided into three parts for training (60%), verification (20%) and test (20%), which includes 3252, 1086 and 1086 images, respectively. In our track fastener dataset, there are four different types of fasteners, i.e., left normal fasteners, right normal fasteners, abnormal fasteners, and missing fasteners. We use the labeling software called LabelImage to generate ground truth bounding boxes by marking the specific location of fasteners, and transform them into standard data styles similar to the PASCAL VOC [30] and COCO [31] formats for the purpose of training and testing.

The training process includes two steps: a coarse stage and the fine-tuning one. In the coarse stage, this paper sets 400 epochs for model training, and the initial and last learning rate is 0.01 and 0.0001. During the fine-tuning stage, the dataset was retrained for 300 epochs with the learning rate of 0.0001 and the last learning rate of 0.00001. The stochastic optimization algorithm proposed by Adam is used to update the network parameters.

Based on the track fastener dataset, we compare the detection performance of different object detection networks on GPU platform. The comparison between YOLOv2 and YOLO [32], Fast R-CNN [33], Faster R-CNN [34], Mask R-CNN [35], SSD [36] and other models is shown in Table 2. It can be seen from Table 2 that the YOLOv2 object detection network superior to other object detection networks in both detection accuracy and detection speed.

5.2. HLS results and analysis

For verify the advantages of the proposed methods, we use Vivado HLS 2017.4 to generate the synthesis report. The resources utilization of the proposed implementations is reported in Table 2, where the non-optimized implementation contains architectures without using optimization methods. In contrast, the optimized implementation uses both parallel implementation, cycle optimization, array partition, ping-pong buffer and multi-channel transmission pragmas to reduce the latency of the design. It can be seen from Table 2 that the optimized resource consumption is slightly greater than the non-optimized. The resources of the BRAM, DSP, FF and LUT by optimized implementation is 1%, 9%, 7%, and 15% higher than that of the non-optimized (see Table 3).

In order to verify the advantages of FPGA hardware acceleration, we verify three implementations of our fastener detection system: (1) SW-based approach (2) HW/SW Co-Design (3) HW-based approach. Several optimizations are applied for the different approaches. Table 4 summarizes the results obtained from the three different approaches.

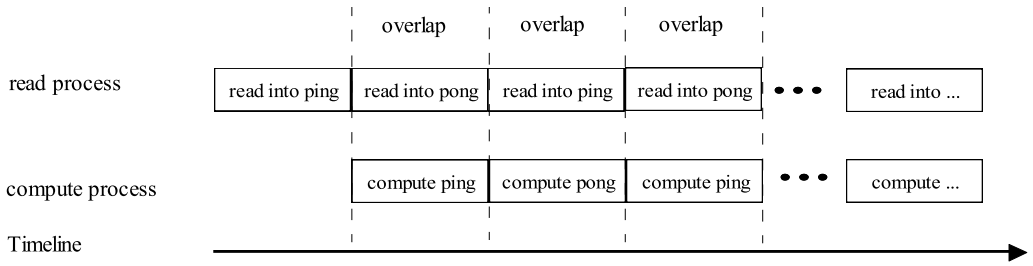
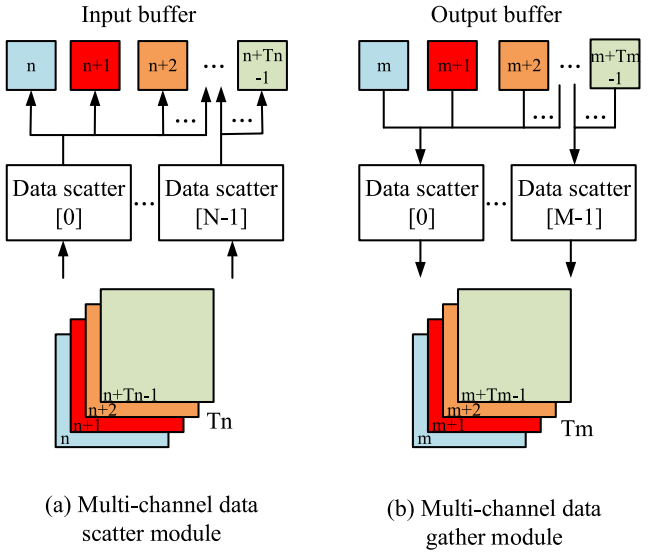


Fig. 10. The timing diagram of ping pong buffer.



(a) Multi-channel data scatter module

(b) Multi-channel data gather module

Fig. 11. Multi-channel data transmission.

Table 3
FPGA resource consumption.

Resources		BRAM	DSP	FF	LUT
Non-optimized	Used	265	148	27 892	53 329
	Available	1510	2020	554 800	277 400
	Utilization (%)	17	7	5	19
Optimized	Used	274	326	71 032	95 244
	Available	1510	2020	554 800	277 400
	Utilization (%)	18	16	12	34

Table 4
Processing time for the three presented implementations.

	Optimizations	Time (s)	Precision
SW-based approach	–	1.37	99.3%
HW/SW Co-Design	–	2.63	98.9%
	Layer parallel implementation	0.27	98.9%
HW-based approach	Pipeline, array partition, ping-pong, multi channel	0.041	98.8%

From the Table 4, we can conclude that the FPGA accelerators provides 9.7× acceleration in the HW/SW Co-Design. On this basis, the HW-based approach provides a faster acceleration of 6.6× compared to the HW/SW Co-Design. Therefore, both HW/SW Co-Design and HW-based approach are far better than SW-based approach in terms of detection speed. In addition, the accuracy of the three approach is 99.3%, 98.9% and 98.8%, respectively. The detection accuracy of HW/SW Co-Design and HW-based approach is low. This is caused by precision parameters of the hardware approach, such as integer, floats, etc. For the fastener detection system we proposed, since the vehicle needs to realize real-time detection at high speed, the detection speed needs to be fast. Compared with the three different approaches, only the hardware approach can meet our requirements for real-time monitoring, and the decline of its accuracy is within an acceptable range.

Table 5
The inference speed under different platforms.

Development platform	Acceleration scheme	Inference time (ms)
CPU	–	986
Pytorch on GPU	–	86
Zynq	HW-based approach	**41**

In addition, we deployed YOLOv2 model in different platforms, including Zynq, Pytorch and CPU. The corresponding inference time is shown in Table 5. Specifically, when tested on a CPU platform, the inference time is 986 ms, which is 1.01 FPS. When tested under the Pytorch platform, the inference time is 86 ms, i.e., 11.63 FPS. When tested on the Zynq embedded platform, the inference time of the hardware-based method is 24 times faster than that of the CPU, and its inference time is 41 ms, which is about 24.39 FPS. In conclusion, our research shows that the real-time fastener detection model on Zynq embedded platform can achieve the faster detection speed.

Fig. 12 shows the track fastener detection results of the YOLOv2 network. For the detection performance, we evaluate it from the following two aspects: (i) fastener position annotations and (ii) confidence level. For the fastener position annotation, we found that the YOLOv2 network can accurately locate the fastener positions in six images. For the confidence level, after training on a large number of fastener images, we found that the confidence level of the YOLOv2 network on the normal fasteners reaches to higher than 80%. However, the confidence level of the abnormal fasteners and missing fasteners was relatively low. Even so, their confidence level is still more than 70%.

5.3. Comparison of different detection algorithms

The YOLOv2 object detection network was put into practical application and tested in our fastener detection system. In addition, we also deployed some other object detection methods through Vivado HLS. In this section, YOLOv2 is compared with traditional image feature extraction algorithms such as HOG [37] and Local Binary Pattern (LBP) [38] through P–R curves to illustrate the superiority of the YOLOv2 object detection model. The P–R curves of fasteners detection using different detection algorithms are displayed in Fig. 13.

It can be seen from Fig. 13 that compared to traditional image feature extraction algorithms, the recall and precision of YOLOv2 have better performance. In addition, the performance of YOLOv2 in precision is better than HOG, LBP and Haar image feature extraction algorithms. Besides, YOLOv2 object detection network has better adaptability under complex working conditions. Therefore, it is indicated that YOLOv2 object detection network has a good effect on improving the detecting precision of fastener in the case of ensuring a high recall rate.

In track fastener detection task, two issues need to be paid long-term attention, that is, missed and mistaken detections. The relevant

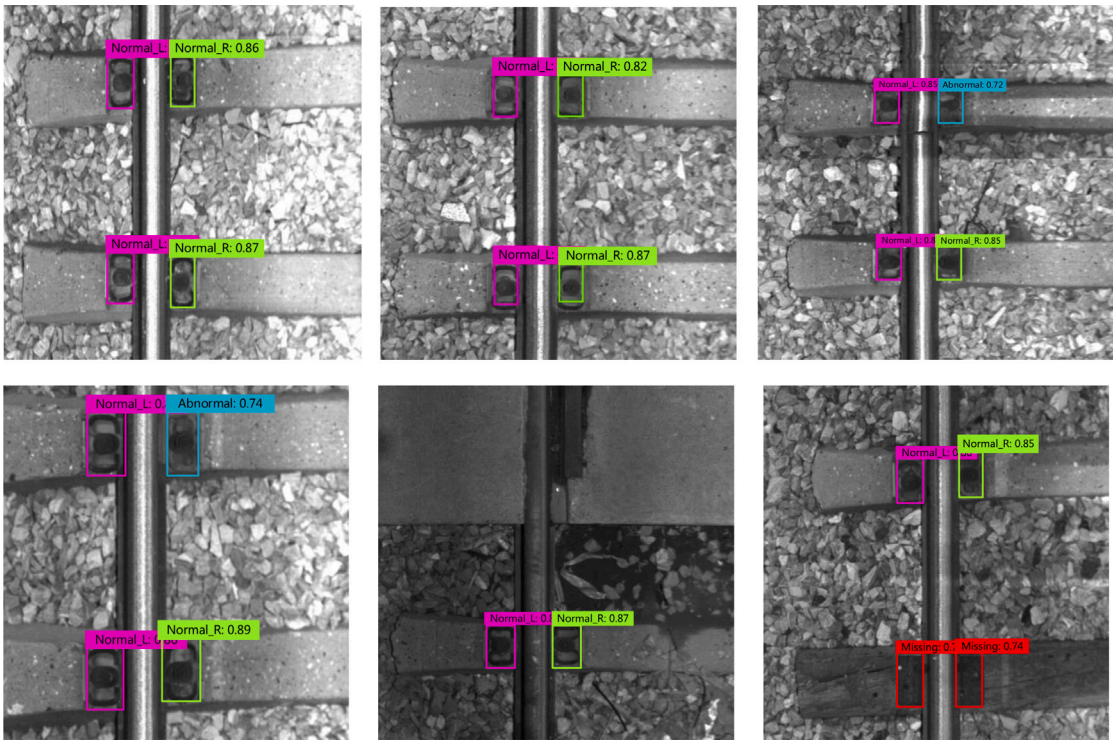


Fig. 12. The track fastener detection results.

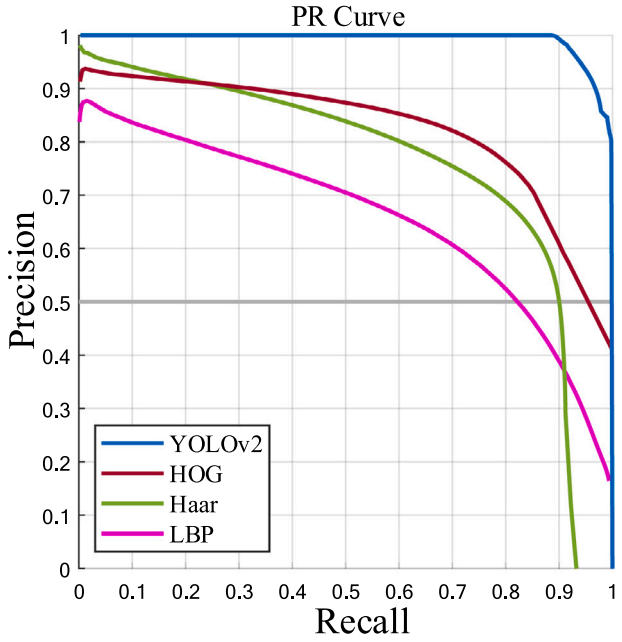


Fig. 13. P–R curves of different detection algorithms.

prediction results are shown in Fig. 14. From the prediction results, it can be seen that the traditional image feature extraction algorithms have obvious missed and mistaken detections. In contrast, the optimized YOLOv2 deployed on Zynq can not only achieve low missed and mistaken detections rate, but also achieve a good detection effect on the fasteners located at the edge of the image.

Table 6 presents mAP, time complexity and space complexity of different algorithms for fasteners detection. From this table, we can see that the mAP of YOLOv2 object detection model is significantly

Table 6
Evaluation index of fastener detection.

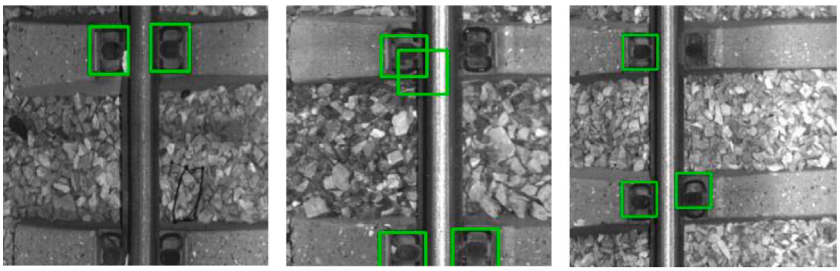
Algorithm	Input size	Inference time (ms)	mAP (%)	F1-score
HOG	4096 * 4096	25	82.3%	0.765
LBP	4096 * 4096	22	64.6%	0.563
Haar	4096 * 4096	24	75.4%	0.672
YOLOv2	4096 * 4096	41	98.8%	0.929

higher than the traditional image processing algorithms. Although inference time of YOLOv2 is somewhat low compared to traditional image processing algorithms, it obtains compensation from high mAP and meets the real-time requirement for real-world applications.

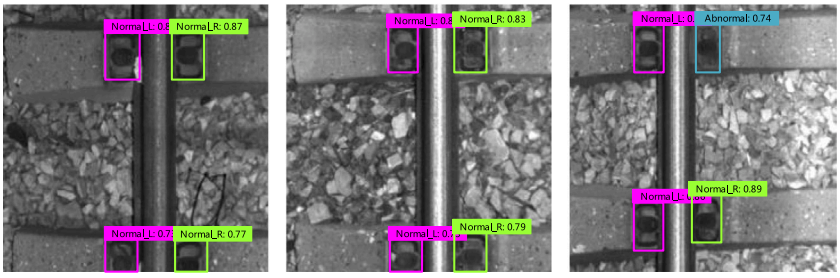
In summary, the extensive results show that, compared with the traditional image feature extraction algorithms, the YOLOv2 object detection model greatly improves the accuracy of fastener detection. Compared with before optimization, the improved YOLOv2 has lower time and space complexity, which lays the foundation for real-time detection of rail fastener under the requirement of high detection accuracy.

6. Conclusion

In this paper, we deployed the YOLOv2 network through HW-based approach to realize the real-time detection of track fasteners. The proposed Zynq SoC implementation not only has high detection accuracy, but also has less resource consumption. The results show that the implementation provides users with higher flexibility in terms of various interfaces, hardware processors and PL. Results presented have shown that the proposed implementation only needs less than 35% of hardware resources to realize the detection of a track fastener image in 41 ms, which outperforms the existing track fasteners detection system in different key performance metrics. In addition, compared with traditional image feature extraction algorithms, the detection accuracy of YOLOv2 has been greatly improved. Therefore, the proposed method satisfies the real-time requirement for track fastener detection



(a)The detection effect of traditional image feature extraction algorithms



(b)The detection effect of YOLOv2 network

Fig. 14. The prediction results of different algorithms.

in track inspection systems, and can better adapt to complex working conditions. The future work will focus on the accuracy optimization of object detection network and real-time detection of fasteners in complex environments.

Data availability

Data will be made available on request

Acknowledgment

This work was supported by Beijing Natural Scicence Foundation (L221017).

Declaration of competing interest

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

References

[1] L.Q. Jin, L.I. Hong-Sheng, Z.L. Liu, H.N. Xing, L.I. Da, L.I. Shu-Hui, Influence of the interaction between frozen-thaw cycles and train dynamic loads on the high-speed railway subgrade settlement, J. Railw. Eng. Soc. (2017) doi:10.3969(2017)11-0014-04.

[2] Q. Li, S. Ren, A real-time visual inspection system for discrete surface defects of rail heads, IEEE Trans. Instrum. Meas. 61 (8) (2012) 2189–2199, http://dx.doi.org/10.1109/TIM.2012.2184959.

[3] A. Hq, A. Tx, B. Gw, C.C. Yu, C.A. Cong, MYOLOv3-tiny: A new convolutional neural network architecture for real-time detection of track fasteners, Comput. Ind. 123, http://dx.doi.org/10.1016/j.compind.2020.103303.

[4] L. Ying, H. Trinh, N. Haas, C. Otto, S. Pankanti, Rail component detection, optimization, and assessment for automatic rail track inspection, IEEE Trans. Intell. Transp. Syst. 15 (2) (2014) 760–770, http://dx.doi.org/10.1109/TITS.2013.2287155.

[5] Q. Feng, W. Qiang, Track detection technology and testing equipment, Urban Mass Transit 10 (3) (2007) http://dx.doi.org/10.3969/j.issn.1007-869X.2007.03.007.

[6] X. Wei, D. Wei, D. Suo, L. Jia, Y. Li, Multi-target defect identification for railway track line based on image processing and improved YOLOv3 model, IEEE Access 8 (2020) 61973–61988, http://dx.doi.org/10.1109/ACCESS.2020.2984264.

[7] G.D. Ruvo, P.D. Ruvo, F. Marino, G. Mastronardi, E. Stella, A FPGA-based architecture for automatic hexagonal bolts detection in railway maintenance, in: Computer Architecture for Machine Perception, 2005. CAMP 2005. Proceedings. Seventh International Workshop on, 2005, 2005, http://dx.doi.org/10.1109/CAMP.2005.4.

[8] M. Singh, S. Singh, J. Jaiswal, J. Hempshall, Autonomous rail track inspection using vision based system, in: IEEE International Conference on Computational Intelligence for Homeland Security & Personal Safety, 2007, http://dx.doi.org/10.1109/CIHSPS.2006.313313.

[9] J. Yang, W. Tao, M. Liu, Y. Zhang, H. Zhang, H. Zhao, An efficient direction field-based method for the detection of fasteners on high-speed railways, Sensors 11 (8) (2011) 7364–7381, http://dx.doi.org/10.3390/s110807364.

[10] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi, L. Chen, Automatic fastener classification and defect detection in vision-based railway inspection systems, IEEE Trans. Instrum. Meas. 63 (4) (2014) 877–888, http://dx.doi.org/10.1109/TIM.2013.2283741.

[11] Q. Dong, A. Wu, N. Dong, W. Feng, S. Wu, A convolution neural network for parts recognition using data augmentation, in: 2018 13th World Congress on Intelligent Control and Automation, WCICA, 2018, http://dx.doi.org/10.1109/WCICA.2018.8630451.

[12] J. Huang, Y. Tian, S. Wang, X. Zhao, S. Peng, Learning visual similarity for inspecting defective railway fasteners, IEEE Sens. J. 19 (16) (2019) 6844–6857, http://dx.doi.org/10.1109/JSEN.2019.2911015.

[13] J. Chen, Z. Liu, H. Wang, A.N. nez, Z. Han, Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network, IEEE Trans. Instrum. Meas. (2017) http://dx.doi.org/10.1109/TIM.2017.2775345.

[14] Y. Li, Z. Han, H. Xu, L. Liu, K. Zhang, YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions, Appl. Sci. 9 (18) (2019) 3781, http://dx.doi.org/10.3390/app9183781.

[15] D. Xiao, F. Shan, Z. Li, B.T. Le, X. Li, A target detection model based on improved tiny-Yolov3 under the environment of mining truck, IEEE Access PP (99) (2019) 1, http://dx.doi.org/10.1109/ACCESS.2019.2928603.

[16] J. Redmon, A. Farhadi, YOLO9000: Better, faster, stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[17] I. Xilix, Zynq-7000 all programmable soc: technical reference manual, 2014, http://www.xilinx.com/support/documentation/userguides/ug585-Zynq-7000-TRM.pdf. (Accessed 30-September-2016).

[18] S. Zheng, X. Chai, X. An, L. Li, Railway track gauge inspection method based on computer vision, in: International Conference on Mechatronics & Automation, 2012, http://dx.doi.org/10.1109/ICMA.2012.6284322.

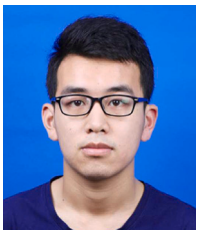
[19] J. Rettkowski, A. Boutros, D. Ghringer, HW/SW co-design of the HOG algorithm on a xilinx zynq SoC, J. Parallel Distrib. Comput. 109 (NOV.) (2017) 50–62, http://dx.doi.org/10.1016/j.jpdc.2017.05.005.

[20] X. Inc., Vivado HLS user guide, 2015.

[21] X. Inc., Vivado design suite user guide: high-level synthesis, 2014.

[22] X. Inc., Vivado design suite user guide: designing IP subsystems using ip integrator, 2014.

[23] J.L. Ortiz, F. Carrió, A. Valero, FPGA implementation of a deep learning algorithm for real-time signal reconstruction in radiation detectors under high pile-up conditions, 2019, doi:arXiv:1903.02439.
[24] A.A. Jarrah, M.M. Jamali, FPGA based architecture of extensive cancellation algorithm (ECA) for passive bistatic radar (PBR), Microprocess. Microsyst. 41 (2016) 56–66, http://dx.doi.org/10.1016/j.micpro.2015.12.003.
[25] A. Ali, H. Djelouat, A. Amira, B. F, M. Benammar, A. Bermak, Electronic nose system on the zynq SoC platform, Microprocess. Microsyst. 53 (2017) 145–156, http://dx.doi.org/10.1016/j.micpro.2017.07.012.
[26] X. Zhai, M. Eslami, E.S. Hussein, M.S. Filali, S.T. Shalaby, A. Amira, F. Bensaali, S. Dakua, J. Abinahed, A. Al-Ansari, Real-time automated image segmentation technique for cerebral aneurysm on reconfigurable system-on-chip., J. Comput. Sci. 27 (JUL.) (2018) 35–45, http://dx.doi.org/10.1016/j.jocs.2018.05.002.
[27] C. Zhang, P. Li, G. Sun, Y. Guan, J. Cong, Optimizing FPGA-based accelerator design for deep convolutional neural networks., in: The 2015 ACM/SIGDA International Symposium, 2015, http://dx.doi.org/10.1145/2684746.2689060.
[28] J. Qiu, S. Song, W. Yu, H. Yang, N. Xu, Going deeper with embedded FPGA platform for convolutional neural network, in: The 2016 ACM/SIGDA International Symposium, 2016, http://dx.doi.org/10.1145/2847263.2847265.
[29] Y. Shen, M. Ferdman, P. Milder, Maximizing CNN accelerator efficiency through resource partitioning, Comput. Archit. News 45 (2) (2017) 535–547, http://dx.doi.org/10.1145/3079856.3080221.
[30] M. Everingham, L.V. Gool, C. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338, http://dx.doi.org/10.1007/s11263-009-0275-4.
[31] T.Y. Lin, M. Maire, S. Belongie, J. Hays, C.L. Zitnick, Microsoft COCO: Common objects in context, in: European Conference on Computer Vision, 2014, http://dx.doi.org/10.1007/978-3-319-10602-1_48.
[32] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, IEEE (2016) http://dx.doi.org/10.1109/CVPR.2016.91.
[33] R. Girshick, Fast R-CNN, Comput. Sci. (2015) http://dx.doi.org/10.1109/ICCV.2015.169.
[34] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1137–1149, http://dx.doi.org/10.1109/TPAMI.2016.2577031.
[35] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, IEEE Trans. Pattern Anal. Mach. Intell. (2017) http://dx.doi.org/10.1109/TPAMI.2018.2844175.
[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, SSD: Single Shot MultiBox Detector, Springer, Cham, 2016, http://dx.doi.org/10.1007/978-3-319-46448-0_2.
[37] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision & Pattern Recognition, 2005, http://dx.doi.org/10.1109/CVPR.2005.177.
[38] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: Application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2037–2041, http://dx.doi.org/10.1109/TPAMI.2006.244.



Tian Xiao received the B.Sc. degree from North University of China Signal and Control of Railway Transit, Taiyuan, China, in 2019; He is currently doing his master's degree with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University. His research interests include deep learning, object detection and image processing.

Tianhua Xu received the B.Sc., M.Sc., and Ph.D. degrees in Electrical and Information Engineering from the Xi'an Jiaotong University, Xi'an, China, Shanghai Jiaotong University, Shanghai, China and Xidian University, Xi'an, China, in 1993, 2002, and 2005, respectively. He is currently a Professor in Beijing Jiaotong University, Beijing, China. His current research interests lie in Multi-Sensor Fusion, Deep Learning, High-speed Image/radar Processing System, Environmental Perception, Advanced Driver Assistance System.

Guang Wang is with the Department of Computer Science, Florida State University, Tallahassee, Florida, USA. He obtained the MS degree at the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China in 2017. Guang is interested in big data analytics, data mining, and machine learning. Currently, his research is focusing on human mobility and charging scheduling of large-scale heterogeneous electric vehicle networks including taxi, bus, car sharing, personal vehicle, etc.