Artificial Intelligence for Engineering Design, Analysis and Manufacturing

www.cambridge.org/aie

Real-time multidimensional detection of longitudinal tears in conveyor belts using FPGA-based parallel acceleration

Fei Li[1], Kun Hu[2,3] and Hangbin Zheng[1]

[1]College of Mechanical Engineering, Donghua University, Shanghai, 201620, China; [2]School of Mechanical Engineering, Anhui University of Science and Technology, Huainan, 232001, China and [3]State Key Laboratory of Mining Response and Disaster Prevention and Control in Deep Coal Mines, Anhui University of Science and Technology, Huainan, 232001, China

Research Article

Cite this article: Li F, Hu K and Zheng H (2024). Real-time multidimensional detection of longitudinal tears in conveyor belts using FPGA-based parallel acceleration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **38**, e19, 1–15
https://doi.org/10.1017/S0890060424000210

Received: 10 January 2024
Revised: 07 July 2024
Accepted: 27 August 2024

Keywords:
conveyor belt longitudinal tear; multidimensional detection; ECSMv3-YOLO; FPGA Accelerator; convolutional acceleration engines

Corresponding author:
Fei Li;
Email: 2698341084@qq.com

Abstract

In the continuous transportation process of coal in mining, exploring real-time detection technology for longitudinal tear of conveyor belts on mobile devices can effectively prevent transport failures. To address the challenges associated with single-dimensional detection, high network complexity, and difficulties in mobile deployment for longitudinal tearing detection in conveyor belts, we have proposed an efficient parallel acceleration method based on field-programmable gate arrays (FPGA) for the ECSMv3-YOLO network, which is an improved version of the you only look once (YOLO) network, enabling multidimensional real-time detection. The FPGA hardware acceleration architecture of the customized network incorporates quantization and pruning methods to further reduce network parameters. The convolutional acceleration engines were specifically designed to optimize the network's inference speed, and the incorporation of dual buffers and multiple direct memory access channels can effectively mitigate data transfer latency. The establishment of a multidimensional longitudinal tear detection experimental device for conveyor belts facilitated FPGA acceleration experiments on ECSMv3-YOLO, resulting in model parameters of 6.257 M, mean average precision of 0.962, power consumption of 3.2 W, and a throughput of 15.56 giga operations per second (GOP/s). By assessing the effects of different networks and varying light intensity, and comparing with CPU, GPU, and different FPGA hardware acceleration platforms, this method demonstrates significant advantages in terms of detection speed, recognition accuracy, power consumption, and energy efficiency. Additionally, it exhibits strong adaptability and interference resilience.

Introduction

In underground mines, continuous material transportation is facilitated by belt conveyors (Ribeiro et al., 2019), operating within challenging environmental conditions (Ji et al., 2020). The coal extracted from the working face may contain sharp angular objects, such as angle steel and gangue. Additionally, height variations exist at the blanking port and the junction between the head and tail of two conveyor belts, heightening the risk of direct punctures or tears in the conveyor belt (Guo et al., 2022; Guo et al., 2022). Longitudinal tearing primarily occurs at the loading and unloading positions of the head and tail, often manifesting as significant cracks or complete tears. These incidents pose potential hazards to transportation, diminishing efficiency, and resulting in economic losses. Hence, investigating real-time detection methods for longitudinal tearing in conveyor belts holds significant importance.

Various methods are employed for detecting longitudinal tears in conveyor belts, including X-ray flaw detection (Xianguo et al., 2018), magnetic detection (Błażej et al., 2018), infrared spectrum (Yang et al., 2020), ultra-high frequency identification (Salim et al., 2021), and laser line scanning (Trybała et al., 2020; Li et al., 2021). However, due to the intricate environment of coal mine transportation and substantial interference, some of these methods may fall short in terms of stability and accuracy. Consequently, the development of conveyor belt longitudinal tear detection utilizing machine vision technology has gained prominence. Traditional machine-learning approaches rely on the manual formulation of features to be learned, followed by feature extraction from images. This approach is associated with challenges such as intricate data preprocessing, slow detection speed, and limited accuracy. In the realm of deep learning, target detection networks primarily fall into two categories: two-stage detection, exemplified by the Fast region-based convolutional neural networks (R-CNN) (Girshick, 2015) and Mask R-CNN (He et al., 2018) series, and one-stage detection algorithms, including the single shot multiBox detector (SSD) (Liu et al., 2016) and YOLO (Redmon et al., 2016) series. Given the high real-time requirements of conveyor belt longitudinal tear detection, one-stage target detection algorithms are predominantly adopted.

Wang et al. (2020) utilized a depth-separable convolution method to reduce the parameters in the SSD algorithm, thereby enhancing its speed. Qu et al. (2020) proposed a conveyor belt damage detection method based on Adaptive Deep Convolutional Network for capturing and classifying damaged targets. However, YOLO, as a real-time object detection algorithm, can directly predict the categories and locations of objects in a single forward pass and is utilized in various detection scenarios. Compared to other models, its main advantages include rapid processing speed, high accuracy, and strong interpretability. Zhang et al. (2022) implemented a Yolov4-based depth-lightweighted target detection network, improving the backbone and neck of Yolov4 to ensure both speed and accuracy in foreign object detection on mining conveyor belts. In a bid to balance accuracy and speed while reducing model complexity, lightweight networks like MobileNet (Howard et al., 2017) and EfficientNet (Koonce, 2021) have been introduced. Zhang et al. (2021) presented an improved Yolov3 algorithm combined with EfficientNet for the simultaneous detection of multiple faults in conveyor belts. Zhang et al. (2021) integrated MobileNet and Yolov4 deeply to achieve lightweight detection, yielding a conveyor belt damage detection accuracy of 93.22%. To enhance network generalization performance, Liu et al. (2022) designed an improved attention mechanism, incorporating it into the image space feature extraction model, resulting in a recognition accuracy improvement of over 20% for the belt damage recognition method.

However, the aforementioned detection methods exclusively focus on the longitudinal tearing of the upper-dimensional surface of the conveyor belt. The lower-dimensional surface of the belt, in contact with the rollers, is susceptible to tearing on its sides or bottom due to sharp objects falling on the roller supports. In terms of network model design, there are dual challenges. Firstly, to enhance detection accuracy and speed, algorithms tend to overlook the increase in the number of parameters. Consequently, the resulting high memory consumption and computational complexity hinder the deployment of edge devices. Secondly, in striving for a lightweight model with improved generalization performance, the design of deep learning network models often neglects a profound integration of features from longitudinal tearing images of conveyor belts and the accelerated convolution calculations at the mobile end.

FPGAs have found application in deploying and accelerating neural networks, leveraging their advantages of low power consumption and high efficiency (Li et al., 2022; Chen et al., 2022; Ganesh et al., 2022). Nguyen et al. (2019) implemented a CNN using the VC707 FPGA, achieving a throughput of 1.877 Megabits per second at a 200 MHz batch. The on-chip power consumption was 18.29 W, and the mean average precision (mAP) of the YOLO network experienced a reduction of 2.63%. Yu et al. (2022) utilized the Xilinx deep neural network development kit to convert an improved YOLOv3 into programmable logic (PL) and deployed it on the Python Productivity for Zynq (PYNQ-Z2) FPGA, achieving a processing speed of 1.54 frames per second (FPS) while consuming a high power of 10 W. Bao et al. (2020) introduced an accelerator design method based on the Winograd algorithm for YOLO, a deep learning object detection model under the PYNQ architecture. Li et al. (2022) applied the Winograd algorithm to optimize the conv3*3 operator in the YOLOv3-tiny network, resulting in improved accelerator efficiency. Adiono et al. (2021) designed a general matrix multiplication (GEMM) processor for the YOLOv3-Tiny network, utilizing an optimal shrink array architecture to conserve on-chip storage space. Yu and Bouganis (2020) pioneered the implementation of a parameterized FPGA custom architecture specifically tailored for YOLOv3-tiny, optimizing it for latency-sensitive applications. Zhang et al. (2022) adopted the static quantization method of fixed-point numbers to achieve a reasoning delay of 498.89 ms for YOLOv4-tiny on ZYNQ-7020, with an average accuracy exceeding 0.95. Xu et al. (2022) designed YOLOv4-tiny convolution and pooling IP kernel on the FPGA platform, accelerating the calculation of convolution and pooling and enabling the identification of coal and gangue.

This paper makes the following main contributions:

(1) Addressing the longitudinal tearing detection issues on both the upper and lower surfaces of conveyor belts, we devised a multidimensional surface target detection test apparatus. This apparatus was designed to gather and analyze multidimensional longitudinal tearing image sample features from the conveyor belts.
(2) Proposing an innovative network model, ECSMv3-YOLO, that embeds a hybrid attention mechanism in the backbone network. This model focuses on the longitudinal tearing features in images under low-light conditions prevalent in coal mines. It features lightweight model parameters, thereby enhancing the recognition accuracy of targets on mobile devices.
(3) Customizing an FPGA accelerator for the ECSMv3-YOLO network, employing fixed-point quantization, model pruning, and normalization of convolution kernel sizes to reduce network computational load. Designing Winograd and GEMM computation engines, utilizing parallelized convolution unfolding to accelerate FPGA processing. To minimize data transfer latency, dual buffers and multiple direct memory access (DMA) channels are introduced.

Finally, FPGA acceleration experiments of the network model were carried out. The results show that this work provides a new method for longitudinal tear detection of mine conveyor belt, and opens the way for low latency object detection on FPGA devices.

Network design and FPGA accelerator

ECSMv3_YOLO network

In the original YOLOv8 algorithms (Terven and Cordova-Esparza, 2023), the extensive use of CSPDarknet network layers resulted in a complex and challenging-to-train model. Considering the need for deploying a lightweight network on FPGA mobile devices, this paper substitutes MobileNetv3 for CSPDarknet as the backbone for feature extraction. To enhance the model's detection capability for longitudinal tears in conveyor belts and its adaptability to various degrees of longitudinal tearing, we introduce a hybrid domain attention mechanism to replace the SENet channel attention mechanism. The improved MobileNetv3 network model, named ECSMv3, depicted in Figure 1, incorporates the ECSNet hybrid domain attention mechanism composed of the efficient channel attention (ECA) (Wang et al., 2020) module and spatial transformer (ST) (Jaderberg et al., 2015) module.

In the ECSMv3 network structure illustrated in Figure 1, the dimension of the input feature layer is initially increased through a 1*1 convolution, followed by feature extraction using 3*3 depthwise separable convolution. Subsequently, an efficient attention module, the ECA module, is constructed using global average pooling and 1D convolution with an adaptive convolution kernel size (k). This module facilitates effective interaction of information between

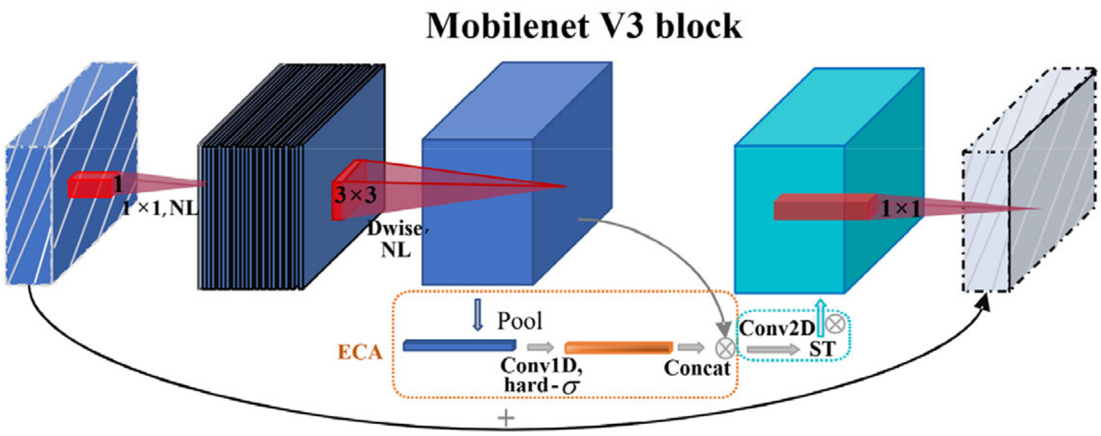

Figure 1. ECSMv3 network structure.

channels, acquiring channel weights to enhance the utilization of pertinent channels. The feature maps of these pertinent channels are obtained and undergo spatial transformation using the ST module and 2D convolution operations, resulting in feature maps post-affine transformation that augment the network's feature extraction capability. Finally, 1*1 convolution is applied for dimension reduction, and the input feature maps are concatenated to obtain the output of the network model, thereby achieving the objective of improving the model's detection accuracy.

The hierarchical parameters of the ECSMv3 network are presented in Table 1. Inspired by the architectural design of the YOLO series networks, a lightweight real-time target detection neural network model, ECSMv3-YOLO, was developed to enhance the model's detection accuracy and inference speed. ECSMv3-YOLO represents an end-to-end object detection framework based on regression, as illustrated in Figure 2.

In the ECSMv3-YOLO model architecture, the K-means algorithm is employed to cluster the dataset and determine the lengths and widths of anchor points. The model's input image size is set to 416 × 416, utilizing the ECSMv3 backbone network for feature extraction. The neck section of the model incorporates a significant number of depth-wise separable convolutional blocks for lightweighting, and the prediction section yields two effective feature layer outputs. Within the SPP structure, max-pooling kernels of sizes 13*13, 9*9, 5*5, and 1*1 are applied to enhance the network's receptive field, facilitating the separation of crucial contextual feature information. The neck section employs a top–down and bottom–up structure for feature fusion, producing two Prediction heads with output sizes of 13 × 13 and 26 × 26, predicting 3 anchors for each scale.

Table 1. Hierarchical parameters of the ECSMv3 network

Hierarchy name	Input	Out	Numbers	Activation function	Attention
Conv2D_BN_ hard-swish	416×416×3	208×208×16	1	hard-swish	No
Bneck_block	208×208×16	208×208×16	1	relu	No
Bneck_block	208×208×16	104×104×24	2	relu	No
Bneck_block	104×104×24	52×52×40	3	relu	Yes
Bneck_block	52×52×40	26×26×112	6	hard-swish	Yes
Bneck_block	26×26×112	13×13×160	3	hard-swish	Yes

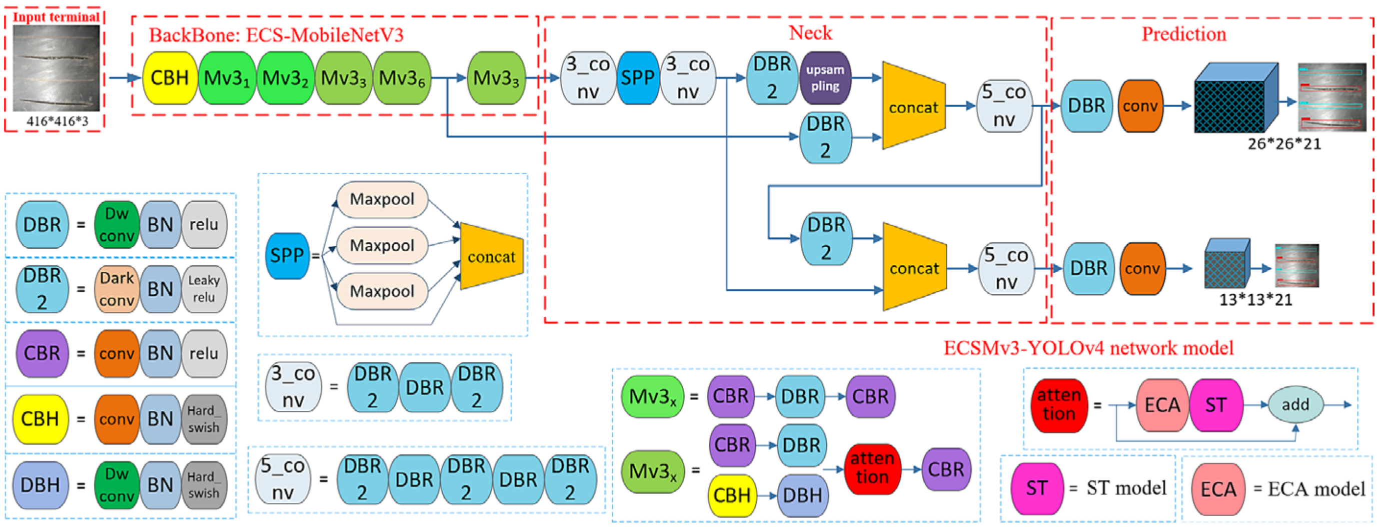

Figure 2. Structure of ECSMv3-YOLO model.

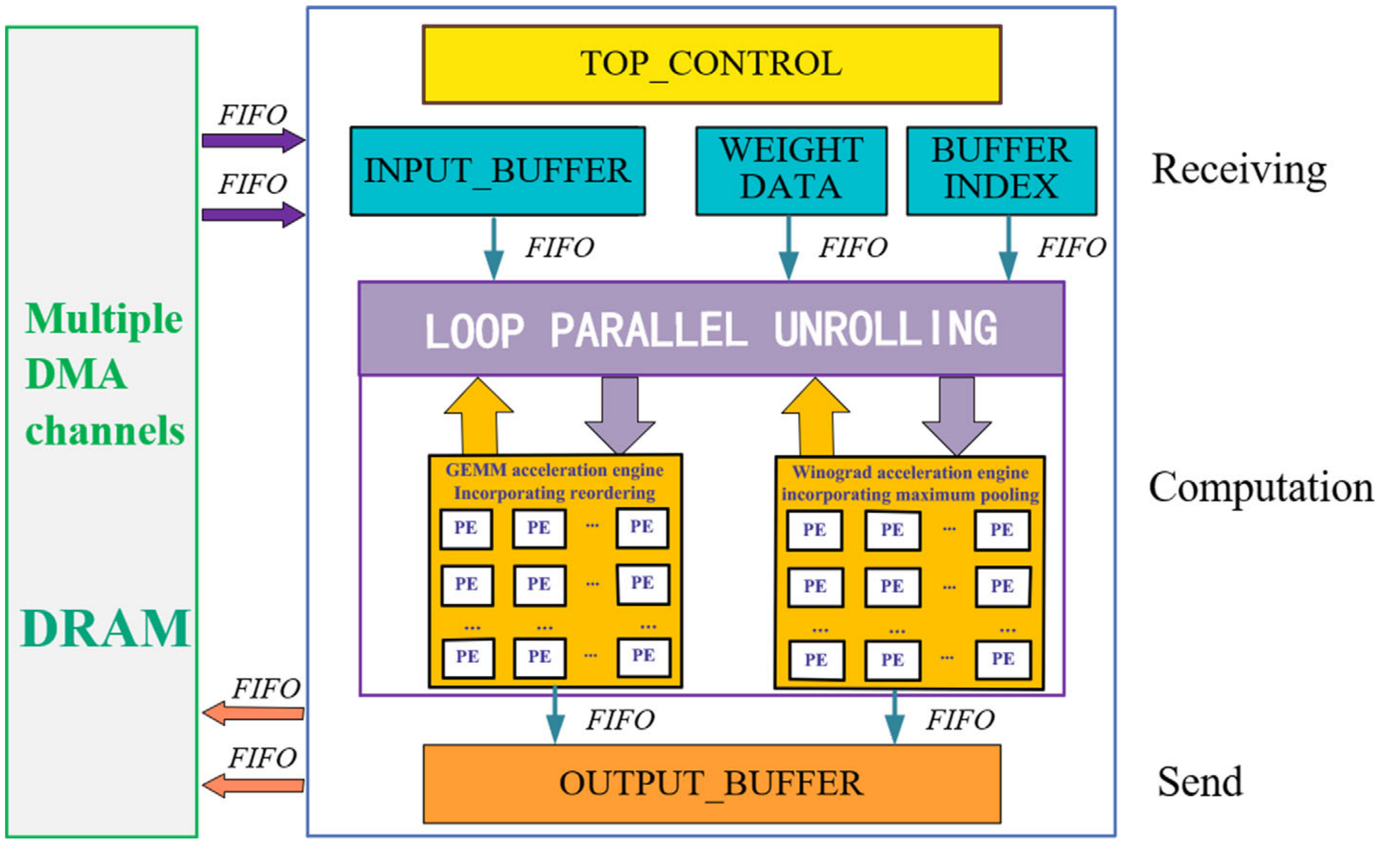

Figure 3. Customized FPGA accelerator architecture for ECSMv3_YOLO.

This ultimately accomplishes the recognition, classification, and output of conveyor belt longitudinal tear images of different classes.

FPGA accelerator

The customized FPGA accelerator for the ECSMv3-YOLO network is depicted in Figure 3, with the top-level control (TOP_CONTROL) executing data transfer. Multiple DMA channels have been designed to leverage off-chip dynamic random access memory (DRAM) resources for caching feature maps, weights, and bias data, while also managing read and write operations to on-chip cache. The data transfer utilizes a first-in, first-out (FIFO) approach, and circular parallel unfolding is employed for convolution computation. To expedite the computations of numerous 3*3 and 1*1 convolutional layers, two engine modules have been devised, switching based on the convolution kernel size, and the feature results are output to the off-chip DRAM cache via FIFO. The entire process involves three stages: data retrieval, convolution computation, and data transmission to the off-chip, facilitating the convolutional layer acceleration of the ECSMv3_YOLO network.

Lightweight FPGA computation network

In the ECSMv3_YOLO network structure depicted in Figure 2, the convolutional kernel comes in five sizes: 13*13, 9*9, 5*5, 3*3, and 1*1. When utilizing FPGA resources for parallel computation of convolutional layers, these five sizes exhibit substantial differences and are not evenly divisible by each other, excluding the 1*1 convolution. During the process of loop parallel convolution unrolling, finding an optimal convolution kernel expansion size suitable for each layer proves challenging, resulting in a wastage of hardware computing resources. To optimize the parallel unfolding computational resources of FPGA convolution in the ECSMv3_YOLO network, a method of convolution kernel size normalization is proposed. This method employs a multilevel concatenation of small-size convolution kernels to substitute for large-size convolution kernels, and its optimization process is illustrated in Figure 4a.

From Figure 4a, it can be observed that replacing the 5*5 convolution kernel with a 2-level 3*3 convolution kernel calculation results in a reduction of convolution weights from 25 to 18. Similarly, the concatenation of 6-level and 4-level 3*3 convolutional kernels optimizes the computation for the 13*13 and 9*9 convolutional kernels, respectively, leading to a decrease in the number of weights from 169 and 81 to 54 and 36. By employing the optimization scheme of convolution kernel size normalization, original convolution kernels larger than 3*3 can be standardized to 3*3. Following the convolutional normalization operation, a significant decrease in the number of convolutional kernel weights is achieved while maintaining the perceptual field of each convolutional layer. This is advantageous for the design of optimal parallel expansion scales in the hardware architecture, further enhancing hardware computational efficiency.

The process of compressing the ECSMv3-YOLO network using pruning techniques is illustrated in Figure 4b. Initially, a scaling factor is introduced for each output channel between the layers of the network model, and it is multiplied by the output weights of each channel. Subsequently, during the training process of the network model, both the network weights and scaling factors are involved in training, simultaneously undergoing sparse regularization. Finally,

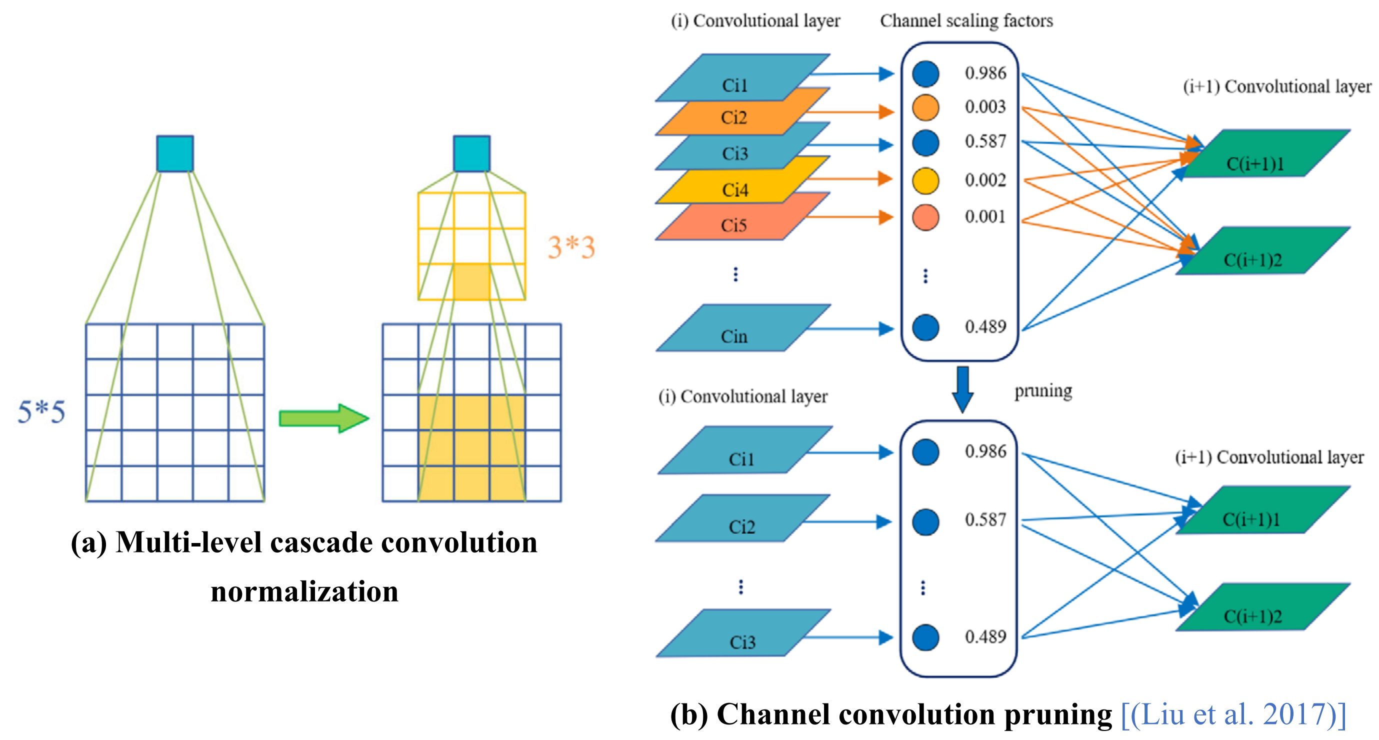
5*5
3*3
(a) Multi-level cascade convolution normalization
(i) Convolutional layer
Channel scaling factors
Ci1
Ci2
Ci3
Ci4
Ci5
Cin
0.986
0.003
0.587
0.002
0.001
0.489
(i+1) Convolutional layer
C(i+1)1
C(i+1)2
pruning
(b) Channel convolution pruning [(Liu et al. 2017)]

Figure 4. Multilevel cascade convolution normalization and model pruning (**b:** Liu et al., 2017).

pruning is applied to channels with smaller-scale factors. The overall training objective of the entire pruning method is as follows:

$$L = \sum_{(x,y)} l(f(x,W),y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \tag{1}$$

Where (x,y) is the training input and the target output. W is the weight of the network model. $f(x,W)$ represents the model function, and l denotes the loss function, which quantifies the discrepancy between the model's predictions $f(x,W)$ and the actual labels y. The network model introduces an L1 regularization term for the scaling factor γ. The first term of L represents loss of model training, the second term of L represents the sum of the penalty $g(\gamma)$ caused by the sparsity of the scaling factors γ, and the two terms are balanced by the regulation factor λ. During the pruning process, when $\lambda < 0.1$, the input and output channels of the convolutional layer are removed from the network.

To effectively utilize FPGA hardware resources and enhance data transfer speed, a dynamic 16-bit fixed-point precision data quantization strategy (Qiu et al., 2016) is employed to quantize the weights, biases, and feature maps of the network model. This helps reduce memory consumption and the utilization of computational resources, allowing for better utilization of external storage bandwidth. The formula for converting 16-bit fixed-point quantized data to signed complement is as follows:

$$V_{\text{fixed16}} = \sum_{i=0}^{15} B_i \cdot 2^{-f_l} \cdot 2^i, \quad B_i \in \{0,1\} \tag{2}$$

Where f_l is the length of the decimal place, $(15 - f_l)$ is the length of the integer bit, combined with the 1-bit part of the symbol to form a complete 16-bit fixed-point number. f_l is the key to dynamic precision quantization. Between different layers and among different feature maps in the convolutional network, dynamic f_l values are employed, resulting in varying lengths of integer and fractional parts for corresponding fixed-point numbers in different layers. Static f_l values are used within the same layer and the same feature map to minimize truncation errors within each layer/feature map. In the convolutional computations of the ECSMv3-YOLO network, shift operations are performed based on the optimal quantization bit-width for each layer. After quantizing the weights, biases, and feature maps, the ReLU activation function is also subjected to 16-bit fixed-point quantization.

FPGA acceleration strategy

Convolution parallel unrolling

The convolutional computation of the proposed ECSMv3_YOLO network comprises four cyclic loops, as depicted in Figure 5. These loops include the convolution kernel loop, input channel loop, input feature map loop, and output channel loop. The convolutional operations are cyclically executed along four loops, sliding between the convolution kernel and feature maps, providing significant parallel optimization space. The key is to design the scale, dimensions, and magnitude for the cyclic parallel expansion within the ECSMv3_YOLO network.

To effectively map and execute convolution loops, loop unrolling is employed to expedite convolution computations, enabling high-throughput and efficient data reuse within the architecture for convolutional calculations. The unrolling of various convolution loops results in distinct parallelization of computations, influencing the optimal processing element (PE) architecture concerning data reuse opportunities and memory access patterns (Ma et al., 2018). Given the FPGA's constrained number of multipliers, this paper aims to enhance pixel data reuse and PE efficiency during FPGA hardware design. It also seeks to balance the impact of different dimensions of unrolling on buffer bandwidth and the depth of intermediate

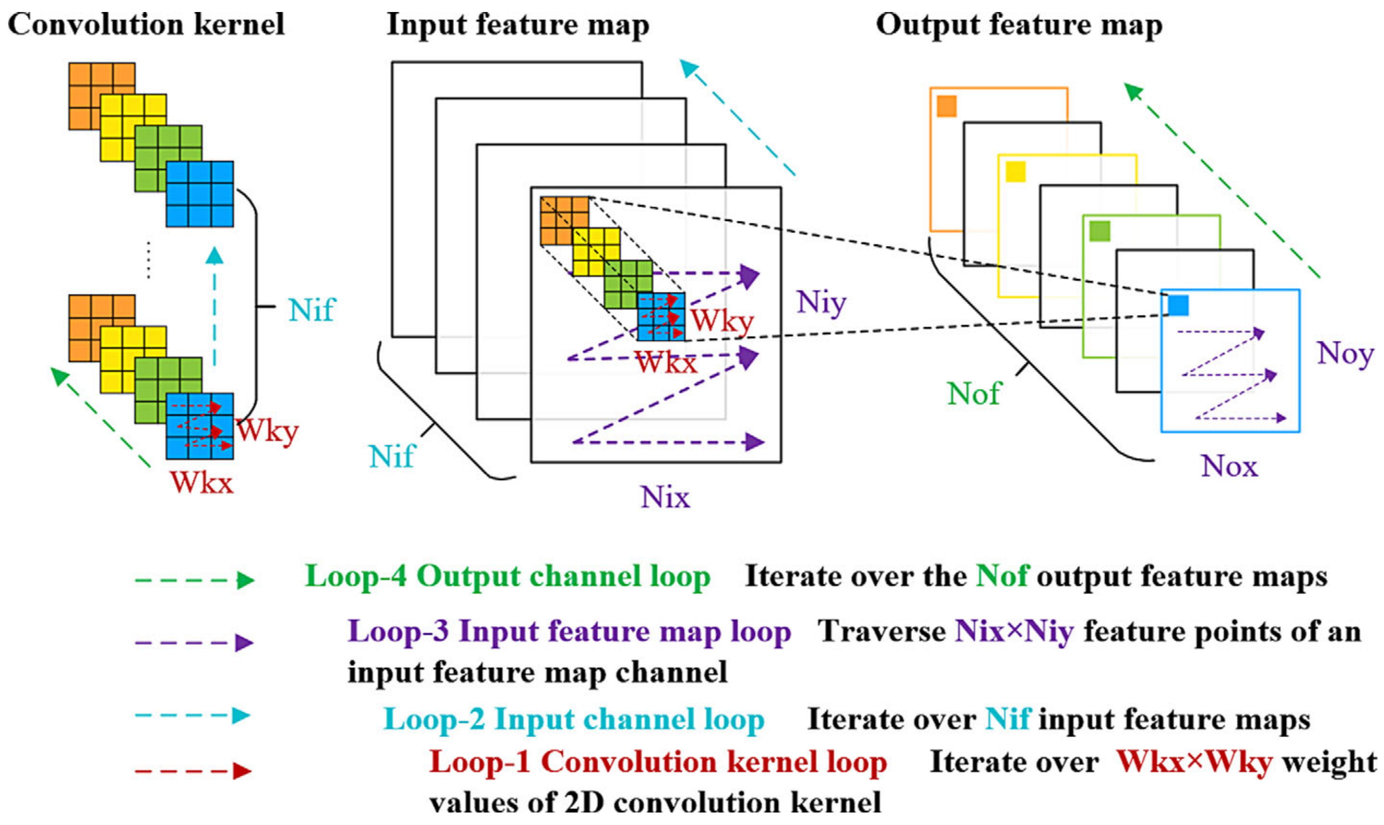
Convolution kernel
Input feature map
Output feature map
Nif
Wky
Wkx
Niy
Nix
Nof
Noy
Nox
Loop-4 Output channel loop Iterate over the Nof output feature maps
Loop-3 Input feature map loop Traverse Nix×Niy feature points of an input feature map channel
Loop-2 Input channel loop Iterate over Nif input feature maps
Loop-1 Convolution kernel loop Iterate over Wkx×Wky weight values of 2D convolution kernel

Figure 5. Four-layer Loops of convolution computation.

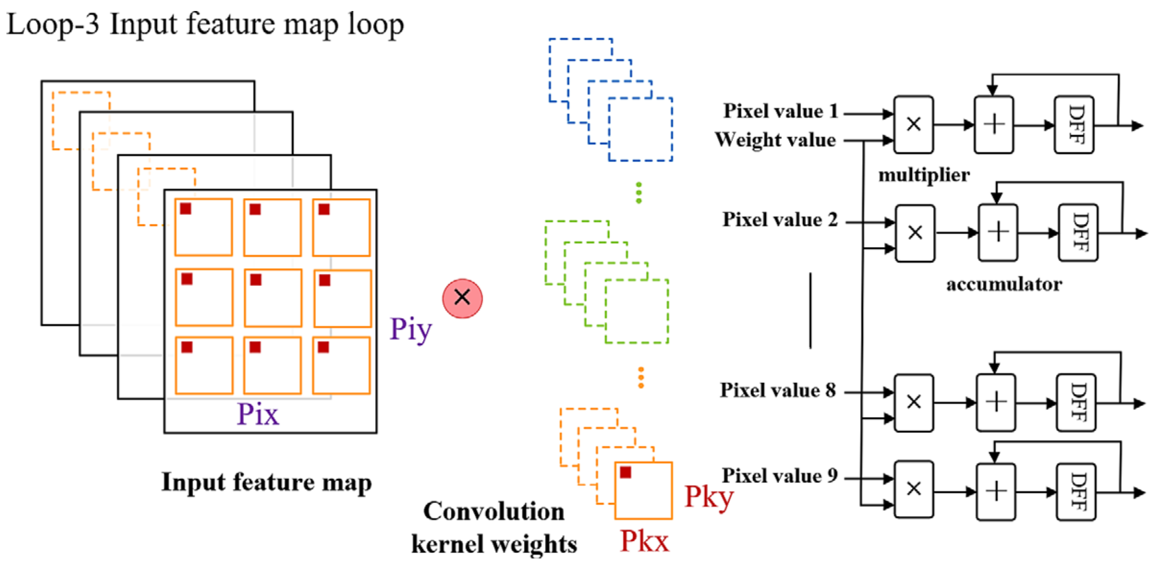

Figure 6. FPGA parallel loop unrolling computation of ECSMv3_YOLO input feature maps (Ma et al., 2018).

results. Specifically, this study explores the loop parallel unrolling of the Loop-3 input feature map in the ECSMv3_YOLO network structure. The hardware unfolding of the design is illustrated in Figure 6.

To maximize the parallelism following loop unrolling, the parallel unrolling scale of different convolutional operation layers should be the greatest common factor of the relevant loops for each layer. The values of loop unrolling variables collectively determine the total number of parallel multiply-accumulate (MAC) operations, calculated using the following formula.

$$Pm = Pkx \times Pky \times Pix \times Piy \times Pof \tag{3}$$

ECSMv3_YOLO network model employs the design approach of convolution kernel size normalization. The size of the convolution kernel $Pkx \times Pky$ of each layer is 3 × 3, the maximum common factor Pof of the input and output channels for each layer is 8, and the maximum common factor $Pix \times Piy$ of the input feature map for each layer is 13 × 13. Therefore, the overall parallelism using the Loop-3 parallel unrolling method is 169.

Design of Winograd engine

For FPGA computation of 2D convolution $F(m \times m, r \times r)$, the number of multiplications for sliding window convolution is $m^2 \times r^2$, while it is reduced to $(m + r - 1)^2$ when employing Winograd convolution. Here, m × m represents the feature map of the ECSMv3_YOLO network, and the convolution kernel size is r × r. The result of the Winograd convolution calculation is expressed as a matrix multiplication, as shown in the following equation (Lu et al., 2017; Lavin and Gray, 2016).

$$Y = A^T\left[\left[GwG^T\right] \odot \left[B^T xB\right]\right]A \tag{4}$$

Where $\odot$ represents the multiplication of the corresponding elements, when F(2 × 2,3 × 3) is calculated, $w = [W_0\, W_1\, W_2]$ is the row element matrix of the 3*3 convolution kernel, $x = [x_0 x_1 x_2 x_3]$ is the row element matrix of the 4 × 4 input feature map, and Y represents the 2 × 2 output feature map obtained after Winograd convolution calculation. A, B, G, and their transpose are the transformation coefficients for Winograd convolution, with their values depicted in the following equation.

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The conversion coefficient matrix includes constants ±1, 0, and ± 1/2, which increase as $(m + r - 1)$ increases. After the convolution calculation parameters are dynamically quantized into 16-bit fixed-point integers. To maintain the recognition accuracy of the proposed ECSMv3_YOLO network, the precision of the convolution kernel must not fall below 2^{-10}, and the value of $(m + r - 1)$ in the Winograd algorithm should not exceed 8. The constant multiplication operation is implemented using shift operations or a combination of 2^m or 2^{-m}, ensuring that it does not consume digital signal processor (DSP) resources.

The PE engine design, incorporating the fusion of Winograd convolution with activation and pooling modules, is depicted in Figure 7. The Winograd convolution module encompasses four key steps. In the first stage, feature maps are acquired from the input buffer, and weights are retrieved from the convolution kernel buffer. These two transformations are independently and concurrently executed, utilizing FPGA's. Look-Up Table (LUT) resources to perform shift operations for constant multiplication in matrices B and G, thus conserving on-chip storage block random access memory (BRAM). In the second stage, three parallel matrices are used to perform dot multiplication on the transformed data blocks of each channel. In the third stage, the output transformation of each channel is derived through Winograd matrix multiplication. In the fourth stage, the resultant values of the output feature maps for each channel are accumulated by using the parallel channel addition tree structure. In the activation pooling module, activation and pooling operations are applied to the obtained single-channel 4 × 4 output feature map. The max pooling kernel size is set at 2*2 with a stride of 2. Finally, three comparators are employed to produce the output results. The integration design of these two modules in the PE engine reduces the transmission latency of convolution and pooling calculations.

According to the pseudocode of the Winograd convolution inner loop in Figure 8, mappings of rows and columns of input features, as well as loop unrolling of input and output channels, can be performed. Different parallelization strategies lead to distinct data sharing and throughput. Considering that parallel unrolling of the row-loop will significantly increase the size of on-chip row buffers, and parallel unrolling of the col-loop may lead to severe memory bank conflicts, the ti-loop and to-loop parallel unrolling methods are employed. Simultaneously, in the convolution layer architecture of the Winograd algorithm on FPGA, the input, output, and convolution kernel buffers are partitioned to maintain

```
for(row = 0; row < H; row += m)              → row-Loop
   for(col = 0; col < W; col += m)            → col-Loop
      for(ti = 0; ti < M; ++ti)                → ti-loop
         for(to = 0; to < N; ++to){             → to-loop
            load: tile_Z(row, col, ti);
            load: filter[ti][to];
            Winograd(tile_Z, filter, tile_Y);
            output(row, col, to) += tile_Y(row, col, ti);
         }
Winograd(w, x, Y){
   $U = GwG^T, V = B^T xB; Y = A^T[U \odot V]A;$
}
```

Figure 8. Pseudocode for the inner loop of the Winograd convolution (Lu et al., 2017).

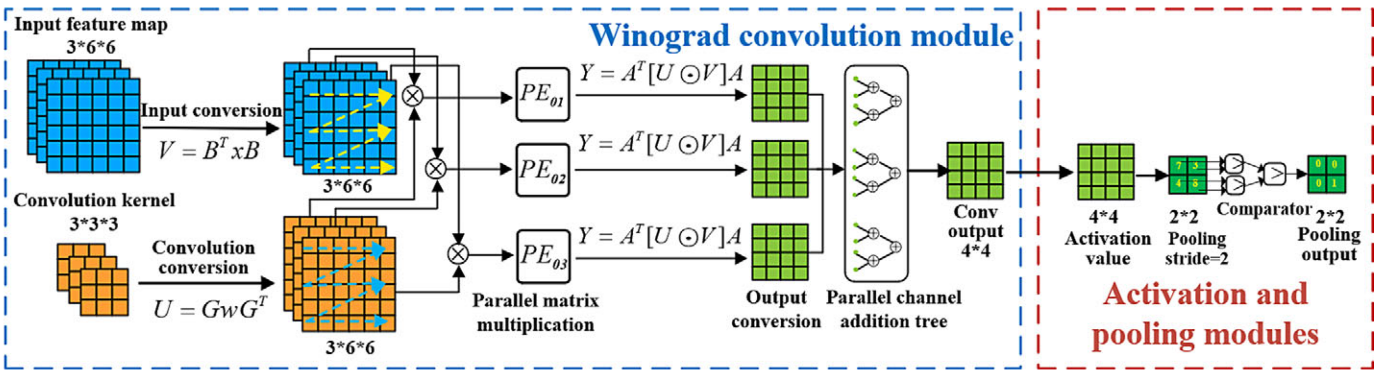

Figure 7. PE engine integrating Winograd convolution, activation, and pooling modules.

effective memory bandwidth. Using Tm and Tn to denote the maximum unfolding factors of ti-Loop and to-Loop, respectively. Based on the different sizes n of input Winograd blocks and the resource consumption of parallelism T_m and T_n, the computational load of DSP consumption is obtained. A regression model is utilized to predict the consumption of LUTs. The formulas for DSP and LUT calculations are as follows:

$$DSP = n^2 \times T_n \times T_m \tag{6}$$

$$LUT = \delta_n^r \times T_n \times T_m \tag{7}$$

Where δ_n^r is the LUT consumption of a single Winograd PE with input Winograd tile size n and convolution kernel size r. Combined with the PE structure of the Winograd convolution and pooling module of the FPGA in Figure 7. The stride between two adjacent Winograd tiles is designed to be m, where $m = n - r + 1$. The amount of BRAM consumed to compute the Winograd convolution is given in the following equation.

$$\text{BRAMs} = r^2 \times T_n \times T_m + (n + m) \times n \times m \times T_m + 2 \times m^2 \times T_n \tag{8}$$

Design of GEMM engine

For the numerous 1*1 convolution operation layers in the ECSMv3 backbone network of the ECSMv3_YOLO model, a PE engine is designed to integrate the GEMM convolution and reordering module, as depicted in Figure 9. The GEMM convolution calculation involves three layers of nested loops. The outer loop mirrors the structure of the 3*3 convolution in the Winograd algorithm. The multiplication and accumulation operations at corresponding positions in the inner loop are transformed into matrix multiplication through the GEMM algorithm. Based on the operation with a convolution kernel size of 1*1, considering the advantage that it eliminates the need for input feature maps to undergo Im2col operations, a series of copying and permutation processes are saved. Utilizing GEMM matrix multiplication for continuous access to the input feature map enhances the efficiency and speed of the 1*1 convolution.

The GEMM module in Figure 9 includes the operation of multichannel feature maps and multichannel 1*1 convolution kernels to obtain the weights of multichannels and all pixel values of the corresponding channel feature maps, the intermediate results are calculated by parallel matrix multiplication, and the output results of single channel are obtained by parallel channel addition tree accumulation. After the output feature map results are obtained by calling the inner loop GEMM algorithm for the [*Nix/Niy*] time, the reordering module is then entered. By designing a 4-way selector, the 2*2 convolution window data on the output feature map is divided into 4 channels, resulting in the output feature map having four times the original number of channels. Consequently, the feature values of each single output channel are half of the output feature map. The fusion of the reordering operation and GEMM convolution calculation module can also be used for the 1*1 convolution layer of the upsampling process in the ECSMv3_YOLO network.

Dual system memory

The dual-cache system and multichannel transmission structure are illustrated in Figure 10, and implemented in FPGA for the ECSMv3_YOLO network to reduce the latency associated with overlapping data loading, data computation, and output stages. It primarily comprises an off-chip dynamic memory DRAM system and a register REG combined with a block buffer BRAM system. During the operation of the convolution layer, the proposed dual-buffer system facilitates the delayed overlap of the three stages: data input, convolution loop unrolling calculation, and result output, allowing for effective data reuse in the DRAM cache. Given that input feature maps are significantly larger than the 3*3 convolution kernel, the convolution kernels are initially unrolled cyclically to enable the reuse of current unrolled feature map pixels and alleviate the buffer throughput pressure on the on-chip BRAM. Subsequently, the input feature maps undergo cyclic unrolling. This process, in conjunction with the feature map blocking strategy, entails caching the feature map first, followed by the on-chip BRAM cache of the weights.

In design of the customized FPGA accelerator for the ECSMv3_YOLO network, a parallel data transmission strategy utilizing multiple DMA channels is employed. Specifically, considering that the number of input channels for the cyclic block of feature maps is 4, and the output channels are 32, the data from each channel of the input feature maps is read in parallel using

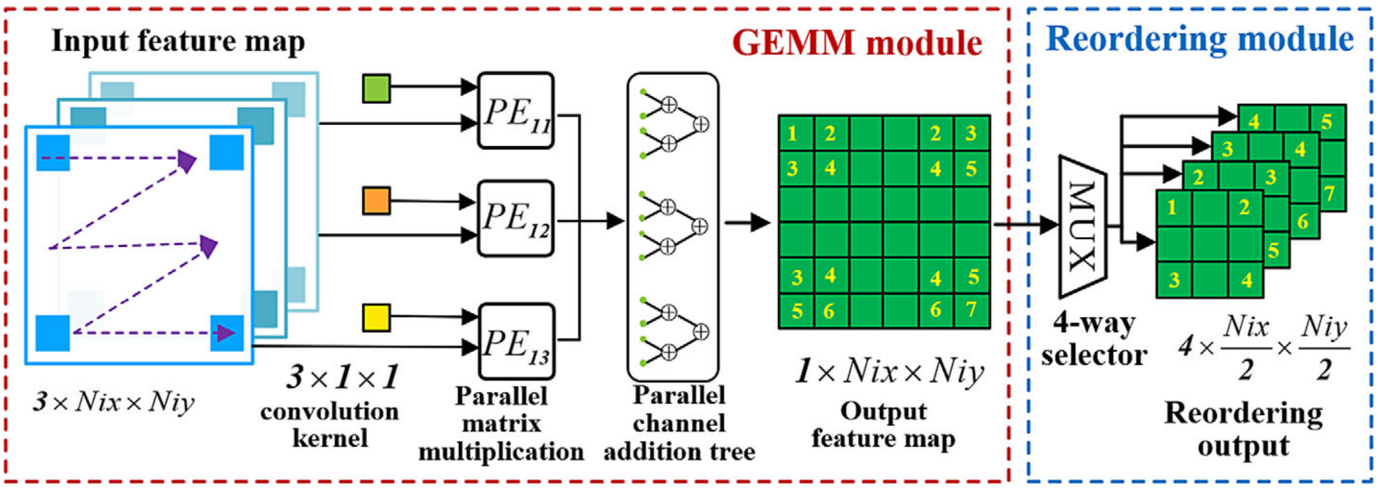
Input feature map
GEMM module
Reordering module
PE_{11}
PE_{12}
PE_{13}
MUX
4-way selector
$3 \times Nix \times Niy$
$3 \times 1 \times 1$ convolution kernel
Parallel matrix multiplication
Parallel channel addition tree
$1 \times Nix \times Niy$ Output feature map
$4 \times \frac{Nix}{2} \times \frac{Niy}{2}$
Reordering output

Figure 9. PE engine integrating GEMM convolution and reordering modules.

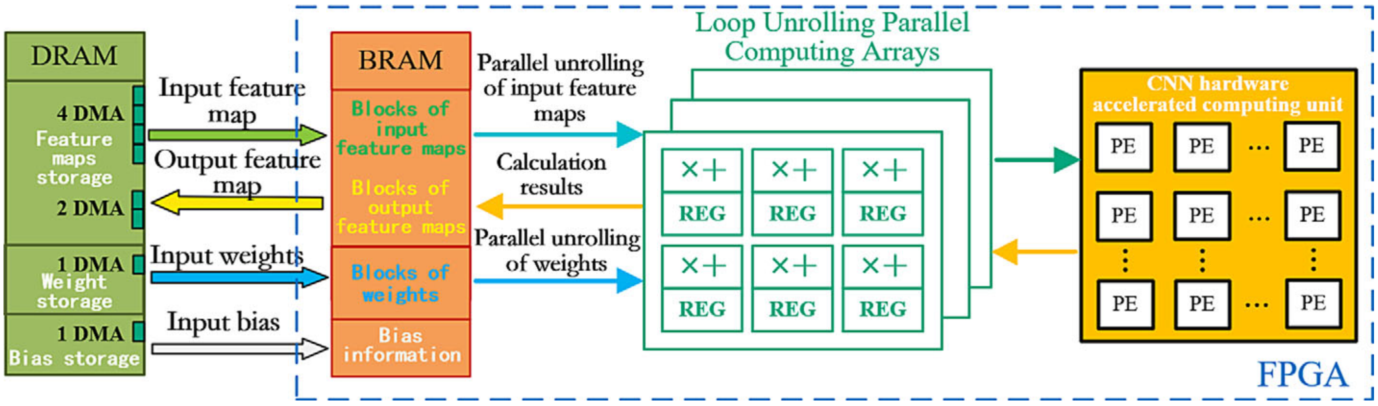
DRAM
4 DMA
Feature maps storage
2 DMA
1 DMA
Weight storage
1 DMA
Bias storage
Input feature map
Output feature map
Input weights
Input bias
BRAM
Blocks of input feature maps
Blocks of output feature maps
Blocks of weights
Bias information
Parallel unrolling of input feature maps
Calculation results
Parallel unrolling of weights
Loop Unrolling Parallel Computing Arrays
×+
REG
CNN hardware accelerated computing unit
PE
FPGA

Figure 10. Dual-system cache and multichannel input architecture.

4 channels. The number of weights of 32 3*3 convolution kernels is less than the 418×418 input feature map pixel values of a single channel, so the weight and bias information are transmitted through the DMA of 1 channel. In consideration of the overlap of the overall delay between pixel loading and feature map processing, the DMA of 2 channels is designed to transmit information for the 32 output channels.

Results and analysis

Experimental equipment

The longitudinal tearing of the conveyor belt is prone to happen not only on the upper surface carrying the coal but also, when the belt deviates or slips, sharp objects in the conveyed coal can easily become lodged in the lower position of the roller, leading to severe tearing of the lower surface of the conveyor belt. Therefore, comprehensive tear detection is essential for both the upper and lower surfaces of the conveyor belt. The designed device for detecting longitudinal tearing in the multidimensional conveyor belt is illustrated in Figure 11. Considering that the head and tail of the belt conveyor are prone to tearing at the loading and unloading points, the upper surface detection camera is suspended directly above the coal-free section in front of the loading port. Simultaneously, the lower surface detection camera is positioned between the upper and lower conveyor belts near the unloading end. This camera setup enables a swift and effective safety inspection of locations prone to tearing. Additionally, it ensures that the collected image samples remain unobstructed by coal, allowing for more comprehensive detection of longitudinal tears on the conveyor belt surface. The designed light sources are arranged to illuminate vertically, and their brightness can be adjusted to simulate varying illumination conditions.

The laboratory-setup multidimensional conveyor belt detection system is illustrated in Figure 12. The conveyor belt utilized is a standard coal mine steel wire rope conveyor belt with a width of 0.6 m and a thickness of 15.0 mm. The belt conveyor is adjustable in speed, with a total length of 4.0 m and a maximum belt speed of 4.0 m/s. Cameras and light sources in the experiment are mounted on a sliding rail, enabling dynamic adjustment of the spacing with the conveyor belt. The camera model is MV-CA003-21UC, a USB3.0 industrial camera designed for visual inspection, with a C-Mount lens and an industrial filter. The light source consists of two FG-DR70-A45-W ring lights, and the light controller is HG-APC2424-C-4CH, used for adjusting the intensity of the two ring lights.

In the process of deploying the ECSMv3_YOLO network for parallel acceleration on FPGA to detect vertical tearing targets, considering that mining conveyor belts work continuously over long distances and frequent shutdowns for maintenance would greatly reduce transportation efficiency, it is therefore crucial to accurately identify and dynamically track the category and location of tearing occurrences. Firstly, when a tear in the conveyor belt is detected, the system responds with signals indicating the type of tear (large crack or complete tear), as well as the position and time captured by industrial cameras. Secondly, based on factors such as continuous conveying speed of the belt, transport distance, position of industrial cameras in both vertical dimensions, occurrence time of tearing events, and downtime duration,

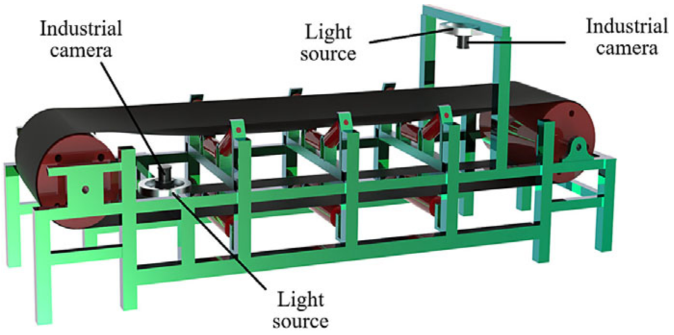

Figure 11. Multidimensional camera and light source placement diagram.

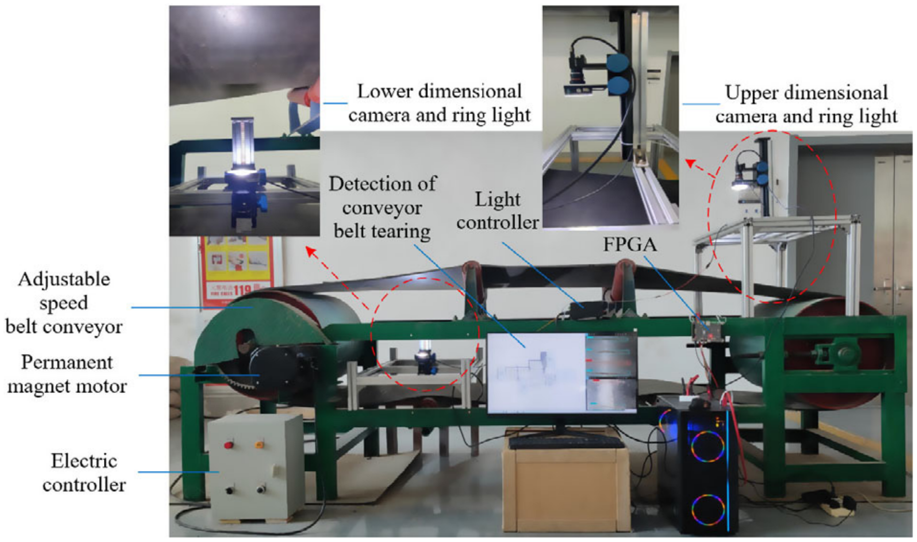
Lower dimensional camera and ring light
Upper dimensional camera and ring light
Detection of conveyor belt tearing
Light controller
FPGA
Adjustable speed belt conveyor
Permanent magnet motor
Electric controller

Figure 12. Experimental device for multidimensional longitudinal tear detection of the conveyor belt.

damaged areas during marking/positioning movements are dynamically calculated to facilitate corresponding repair strategies after shutdown.

ECSMv3_YOLO results

The quality of the dataset significantly influences the detection of longitudinal tears in conveyor belts in actual production environments. Considering that steeply inclined mining conveyors operate at very low speeds (typically 0.5 m/s) to prevent ore spillage, this study conducts experiments on image collection, enhancement, detection performance, and FPGA parallel acceleration testing at a conveyor speed of 0.5 m/s. Utilizing industrial cameras placed at various locations to capture video sources, the video frame extraction method was applied to extract images containing longitudinal tears. A total of 350 image samples were obtained, representing both the upper and lower surfaces of the conveyor belt. To address the issue of overfitting in network training with small-sample data, a variety of augmentation techniques, including mosaic augmentation, affine transformation, rotation shearing, flipping, and adding Gaussian noise, are employed. These techniques enhance the longitudinal tear dataset

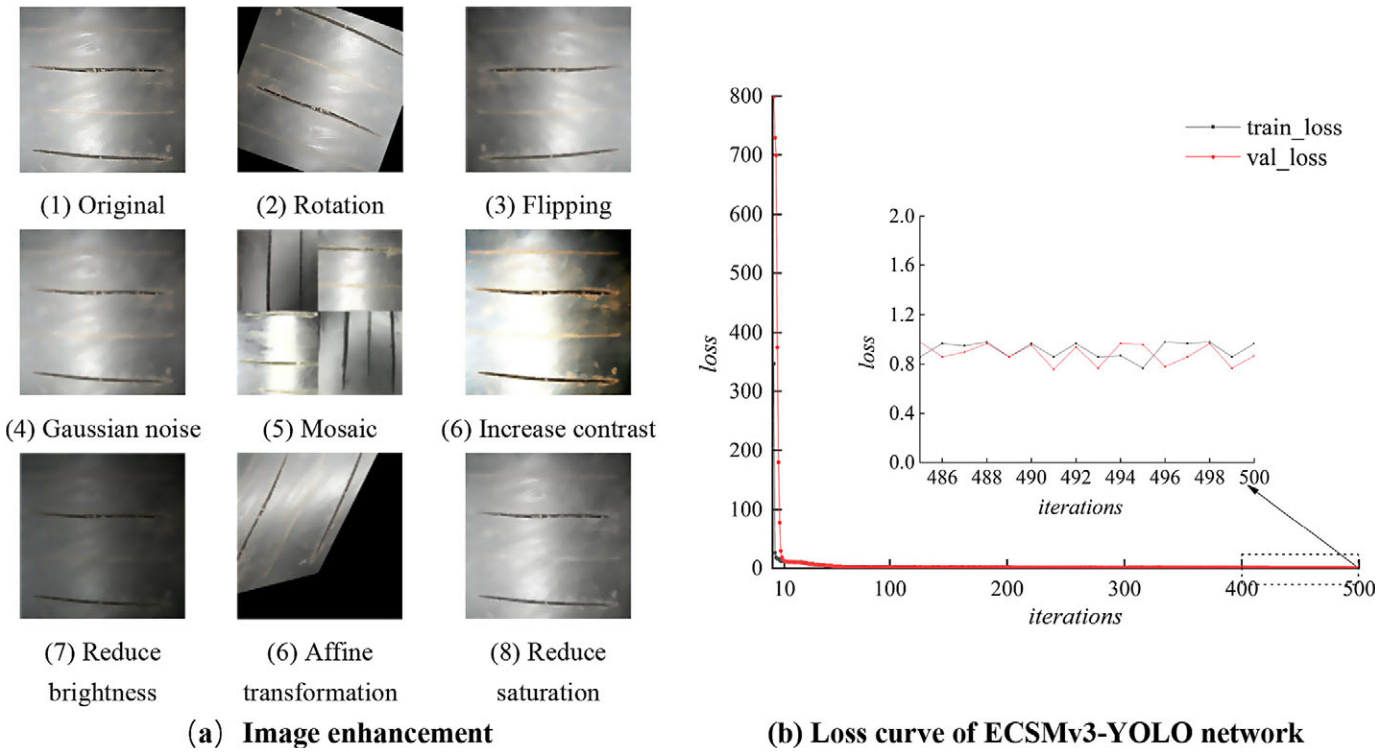

Figure 13. Image enhancement and network training loss.

for conveyor belts to a total of 1500 samples. This enhances the model's ability to discern the positions of image targets, thereby facilitating more efficient feature extraction. The images obtained from the original acquisition and the augmented samples are illustrated in Figure 13a, depicting tears categorized as either large cracks or complete tears.

The primary objective of the ECSMv3_YOLO network is to minimize the computational load and parameters of the model, ensuring a balance between detection accuracy and speed, thereby achieving a design characterized by low complexity and lightweight attributes. The dataset was partitioned using 10-fold cross-validation, where the training and test sets were mutually exclusive with a ratio of 9:1. This procedure was iteratively repeated 10 times for both training and testing. Take the average of these tests' results to comprehensively evaluate the performance of the model. The loss curve of ECSMv3-YOLO trained using the NVIDIA RTX 2070 SUPER GPU is depicted in Figure 13b. During the initial 1–10 iterations, the loss values are relatively high with a sharp decrease. After 500 iterations, the loss value stabilizes around 0.8, indicating the acquisition of favorable model parameters. The model's performance is assessed through metrics such as mAP, FPS, floating point operations (FLOPs), and the total number of training parameters (Params).

The MobileNetv3 network was enhanced by integrating the SENet, CBAM, ECA, and the hybrid domain attention mechanism ECSNet proposed in this paper. The results of different attention mechanisms in the ECSMv3_YOLO network are displayed in Table 2. Through comparative analysis, the addition of the channel domain attention ECA module exhibited a superior effect compared to SENet. With a marginal FPS difference of 1.0 frame/s, the detection accuracy of the hybrid domain attention mechanism ECSNet slightly surpassed that of ECA, and it was 1.3% higher than CBAM. The FLOPs and Params were reduced by 0.003 G and 0.752 M, respectively. The test results confirm that the ECSNet designed in this study demonstrates better performance.

Table 2. Results of changing attention mechanism in the network

Network	Backbone network	FLOPs (G)	Params (M)	mAP	FPS (frames/s)
ECSMv3_YOLO	MobileNetv3 + SENet	7.030	11.309	0.952	33
ECSMv3_YOLO	MobileNetv3 + ECA	7.027	9.797	0.958	35
ECSMv3_YOLO	MobileNetv3 + CBAM	7.032	10.550	0.963	33
ECSMv3_YOLO	ECSMv3	7.029	9.798	0.976	34

Table 3. Effect comparison of different networks

Network	Backbone network	FLOPs (G)	Params (M)	mAP	FPS (frames/s)
YOLOv4-Tiny	CSPdarknet53-Tiny	6.823	5.876	0.875	45
YOLOX-s	Focus+CSPDarknet	11.254	8.938	0.952	35
YOLOv5-s	C3 + CSPDarknet	6.741	7.025	0.945	38
YOLOv8-s	C2f + CSPDarknet	12.104	11.136	0.963	40
ECSMv3_YOLO	ECSMv3	4.882	8.851	0.978	37

The superiority of ECSMv3_YOLO is validated by comparing the performance of different network models, as shown in Table 3. The ECSMv3_YOLO and YOLOv4-Tiny networks both employ two-scale prediction heads and six anchors, with lower FLOPs and Params. YOLOv4-Tiny achieves a maximum FPS of 45 frames/s but with a minimum mAP of 0.875. In comparison to the YOLOX-s network, ECSMv3_YOLO outperforms in various evaluation metrics. In comparison to YOLOv5-s, ECSMv3_YOLO demonstrates a reduction of 1.859G FLOPs and a 3.3% improvement in mAP, with the Params difference of 1.826 M. Furthermore, compared to YOLOv8-s, ECSMv3_YOLO achieves further reductions in FLOPs and Params, reaching 4.882G and 8.851 M, respectively. Despite a 3 frames/s difference in FPS, ECSMv3_YOLO shows a 1.5% improvement in mAP, reaching 0.978. These comparative results indicate the significant advantages of employing ECSMv3_YOLO for multidimensional longitudinal tear detection on conveyor belts.

Different hardware acceleration

The platform for detecting conveyor belt longitudinal tear targets with different hardware accelerations is depicted in Figure 14. In the FPGA acceleration of the ECSMv3_YOLO network, IP cores for pwconv, dwconv, conv, fc, shortcut, and sampling were crafted using Vivado HLS 2019.2. Leveraging the block design module of Vivado 2019.2, the IP cores for each module are instantiated, and the IP core parameters for the ZYNQ development board are configured. The complete IP core architecture is depicted in Figure 15. Test results reveal that the total delay of the hardware system for accelerating the ECSMv3_YOLO network using FPGA is 255 ms, with a power consumption of 3.2 W, achieving a throughput of 15.56 GOP/s and an energy efficiency ratio of 4.86 GOP/s/W.

Within the overall architecture of IP core acceleration for the ECSMv3_YOLO network, the pwconv IP core utilizes GEMM convolution-reranking fusion operations to expedite the 1*1 convolution in pointwise convolution. The dwconv and conv IP cores utilize Winograd convolution-BN-activation-pooling fusion operations to accelerate standard and depthwise separable 3*3 convolution layers. The fc IP core is responsible for accelerating the global average pooling layer and fully connected layer. The shortcut IP core accelerates the residual layer, and the sampling IP core accelerates the sampling layer. The IP cores on the PL side are invoked multiple times by the PS side to achieve the acceleration of the ECSMv3_YOLO network model. Based on GPU-trained network weights, they are stored separately according to the network architecture's point-wise convolution layers, channel-wise convolution layers, residual layers, ordinary convolution layers, sampling layers, and prediction heads. The test dataset samples and network weights are stored in SD cards, and the data is read using the API provided by Xilinx.

The recognition results of the FPGA-accelerated ECSMv3_YOLO network platform, developed using Xilinx Vitis 2019.2, are depicted in Figure 16. A comparison of mAP under different light intensities reveals a slight decrease compared to the 0.978 achieved by the GPU. This discrepancy is primarily attributed to partial accuracy loss resulting from dynamic 16-bit fixed-point quantization and model channel pruning. Similar conclusions are drawn from the light-intensity tests conducted for GPU acceleration. The network demonstrates optimal testing performance under low-light intensity conditions of 84 lux, achieving a remarkable mAP of 0.962, aligning with the dark conveying environments

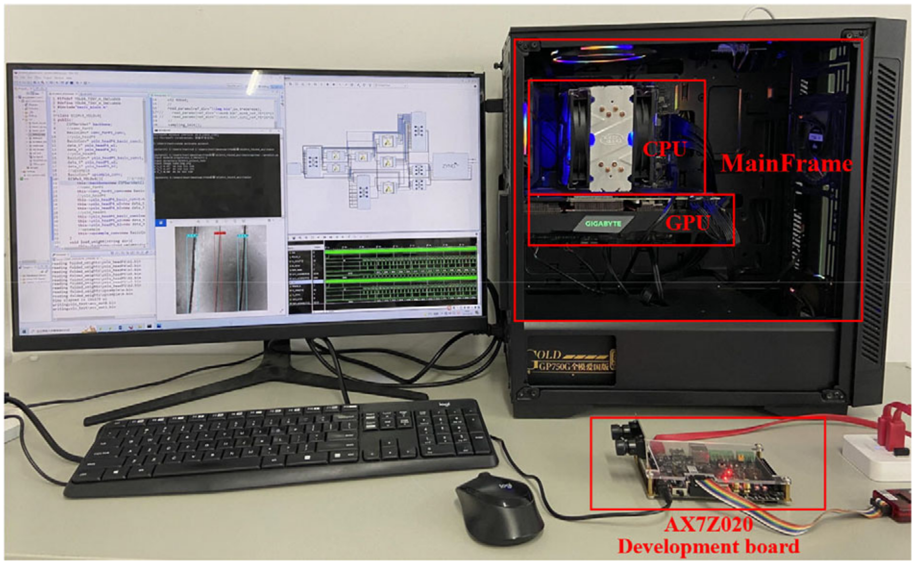

Figure 14. Target detection platform with various hardware accelerations.

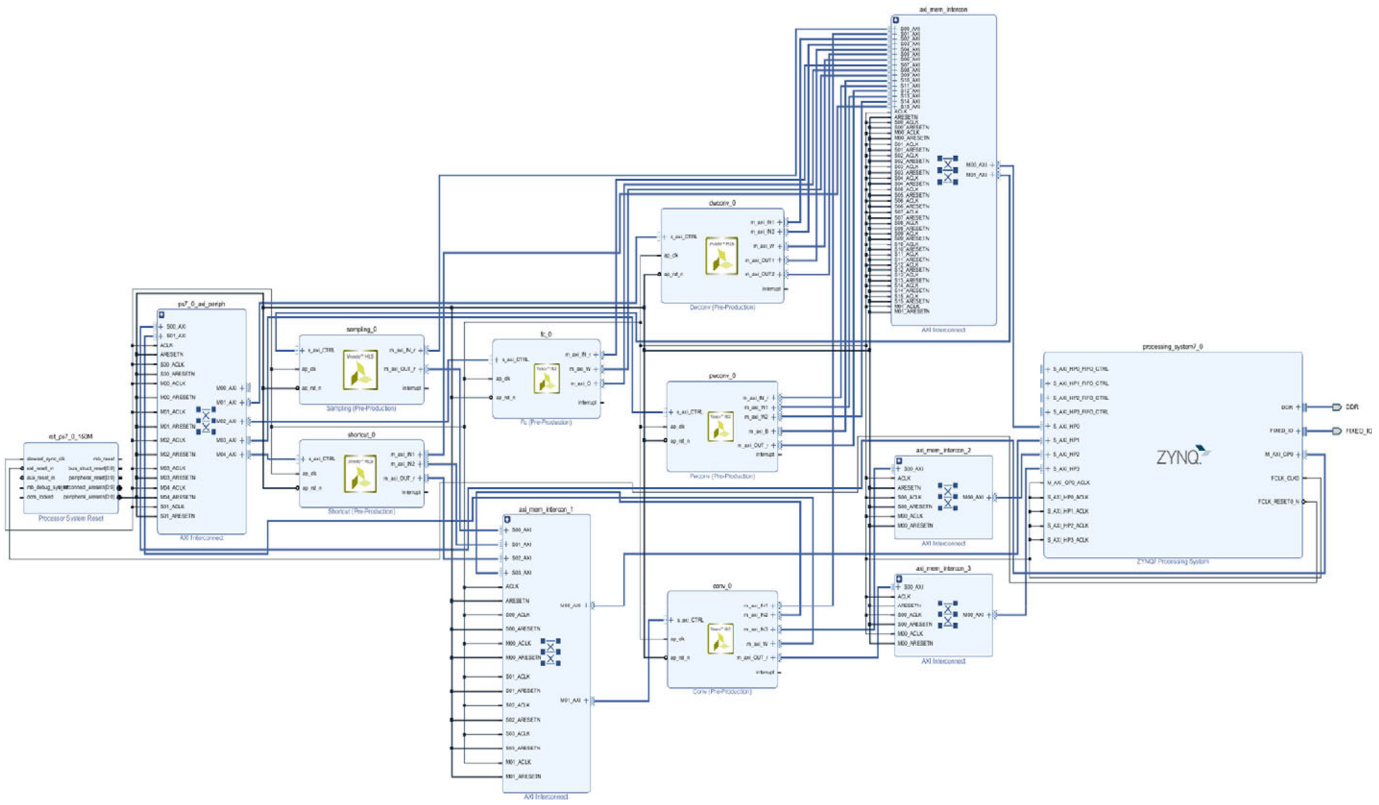

Figure 15. IP cores for FPGA hardware accelerators.

typical in most mines. Even under moderate and strong light backgrounds, the model exhibits high recognition accuracy, with mAP exceeding 0.95, showcasing robustness in longitudinal tear detection.

FPGA accelerator performance analysis

To validate the effects before and after pruning and quantization of the ECSMv3-YOLO network, comparative experimental results are presented in Table 4. The mAP of the ECSMv3-YOLO network, quantified with Fixed-32 before model pruning is 0.978. After model pruning and Fixed-16 quantization, the mAP decreases by 0.016, and the average inference time is reduced from 418 to 255 ms. The number of model parameters is compressed from 8.851 to 6.257 M, and FLOPs are reduced by 18.74%. The results reveal the existence of numerous redundant weight values in the original ECSMv3-YOLO network, and channel pruning effectively reduces model complexity, preventing overfitting. The pruning and

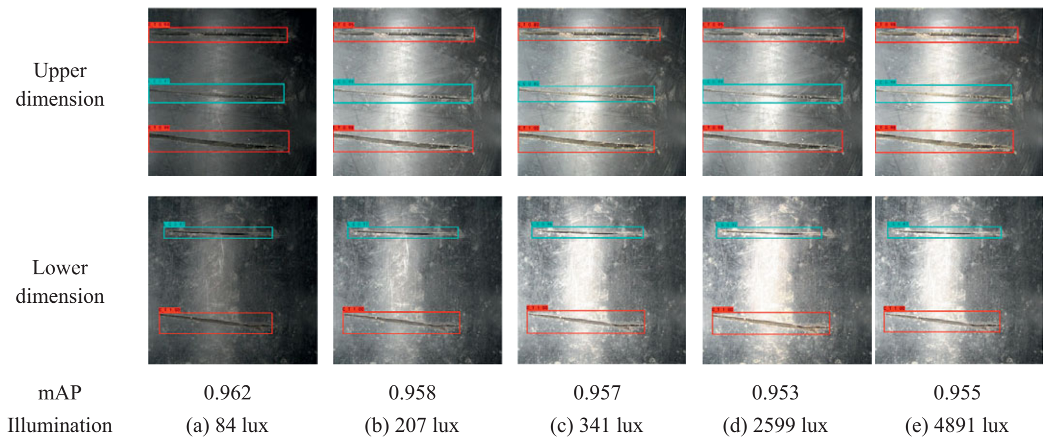

Figure 16. Detection results of FPGA-accelerated ECSMv3_YOLO network.

Table 4. Pruning experiment results of ECSMv3-YOLO network.

Network model	Quantization accuracy	FLOPs (G)	Inference time (ms)	mAP	Params (M)
ECSMv3-YOLO (Before pruning)	Fixed–32	4.882	418	0.978	8.851
ECSMv3-YOLO (After pruning)	Fixed–16	3.967	255	0.962	6.257

Table 5. Comparison results of different acceleration engines.

Acceleration engine type	Frequency (MHz)	Quantization accuracy	Power (W)	Total delay (ms)
Winograd + activation + pooling	150	Fixed–16	3.5	286
GEMM + reordering	150	Fixed–16	3.9	342
Winograd + activation + pooling, GEMM + reordering	150	Fixed–16	3.2	255

quantization methods significantly reduce FPGA inference time, the number of parameters, and the computational workload, thereby enhancing the model's detection speed.

According to the pruned ECSMv3_YOLO network, the size of the large convolution kernels in the network is normalized to 3*3. The experiment is conducted on the FPGA development platform of AX7Z020, and Table 5 presents the performance comparison results of different acceleration engine types. It is observed that when the operating frequency is 150 MHz and dynamic 16-bit fixed-point quantization technology is employed, the performance of the three FPGA accelerator engines is high. Among them, the power consumption and total latency of the GEMM+ reordering engine are higher at 3.9 W and 342 ms, respectively. Due to the characteristics of the ECSMv3_YOLO network, which involves a significant number of depthwise separable convolutions, a greater number of combination operations of 3*3 and 1*1 convolutions are included. Leveraging an engine that integrates Winograd convolution with activation and pooling to accelerate 3*3 convolutional layers, combined with the GEMM algorithm fused with reordering

Table 6. Hardware resource utilization of FPGA.

Resources	BRAM	DSP	FF	LUT
Available resources	280	220	106,400	53,200
Actually using resources	247	215	82,537	43,465
Consumption ratio	88.2%	97.73%	67.6%	81.7%

to accelerate 1*1 convolutional layer, yields a power reduction of 0.4 and 0.7 W compared to either acceleration engine alone. The minimum total latency achieved is 255 ms. The results indicate that utilizing the combined engine for accelerating the ECSMv3_YOLO network in longitudinal tear detection brings noticeable performance advantages.

Simulated testing of FPGA resource utilization for the ECSMv3_YOLO network was conducted in the Vivado HLS 2019.2 environment. For the Zynq-7020 core component, there is a 64-bit AXI_ACP interface on the PL side, 8 DMA channels, 220 DSP slices, 280 slices of memory with a total of 4.9 Mb BRAM, 85 K PL cells, and 53,200 LUT cells. The resource utilization percentages are detailed in Table 6, with the LUT utilization rate reaching 81.7%. Specifically, the on-chip BRAM resources utilize 247 pieces, with BRAM and Flip-Flop (FF) consumption rates of 88.2% and 67.6%, respectively. Efficiently utilizing on-chip cache resources can effectively reduce the frequency of off-chip DRAM accesses, thereby lowering power consumption and FPGA latency. Due to the adoption of the strategy of loop parallelization and unrolling in convolution calculations, a significant number of 3*3 and 1*1 convolutional layers are computed using Winograd and GEMM-based PE engines. Consequently, a higher utilization of on-chip DSP resources is observed, reaching 97.73%.

The ECSMv3_YOLO network was tested on various hardware acceleration platforms, including host CPU, GPU, and AX7Z020, assessing power consumption, total delay, and energy efficiency ratio indicators. The results are presented in Table 7. It is evident that the Intel i7-10700 CPU-accelerated hardware does not outperform in all indicators. In comparison to CPU and GPU hardware acceleration, the AX7Z020 hardware acceleration platform exhibits the lowest power consumption at 3.2 W and a computing time per image of 0.255 s at a frequency of 150 MHz. The FPGA's energy efficiency is 4.834 GOP/s/W higher than that of the CPU and 0.16 GOP/s/W lower than that of the GPU.

Table 7. Comparative results of ECSMv3-YOLO network on different hardware platforms.

Names	CPU	GPU	FPGA
Hardware platform	Intel i7–10700	RTX 2070 SUPER	AX7Z020
Frequency	4.59 GHz	471 MHz	150 MHz
Bit width (bits)	Float–32	Float–32	Fixed–16
Average precision mean (mAP)	0.978	0.978	0.962
Power (W)	48	36	3.2
Computing time per image (s)	3.814	0.027	0.255
Energy efficiency ratio (GOP/s/W)	0.026	5.02	4.86

Overall, the FPGA hardware acceleration architecture demonstrates superior energy efficiency, lower delay, and minimal power consumption, validating the clear hardware advantages of the FPGA acceleration approach for the ECSMv3_YOLO network.

In comparison with the work of other researchers focusing on the FPGA-based acceleration of YOLO series algorithms, the results are presented in Table 8. The findings indicate that, when utilizing similar hardware resources and employing Fixed-16 quantization bit width, this paper achieves higher energy efficiency and throughput while maintaining lower power consumption and total delay compared to the works of Yu et al. (2022), Yu and Bouganis (2020), and Zhang et al. (2022). The FPGA hardware's ability to achieve a higher frequency contributes to improved hardware performance. Adiono et al. (2021) accelerated YOLOv3-tiny using Fixed-8 quantization at a hardware frequency of 250 MHz, achieving further latency reduction and increased energy efficiency. However, in this study, to ensure the average detection accuracy of the ECSMv3_YOLO network, a dynamic Fixed-16 quantization method was employed, resulting in higher FPGA hardware acceleration throughput of 15.56 GOP/s and lower power consumption of 3.2 W.

Conclusion and future work

This paper introduces the innovative ECSMv3_YOLO network, designed for longitudinal tearing detection on conveyor belts. In additrion, FPGA acceleration strategies were implemented to compute the network, reducing edge computing latency and power consumption. A multidimensional detection device for longitudinal tearing on conveyor belts was constructed, and the performance of the network and FPGA accelerator is tested.

(1) Samples of longitudinal tears on the upper and lower dimensions of the conveyor belt were collected under varying light intensities. The backbone network ECSMv3 was constructed, embedding the hybrid attention mechanism ECSNet to extract image features. Using a combination of 2Heads and 6Anchors, the ECSMv3_YOLO network was built, with test results surpassing YOLO series networks, achieving mAP of 0.978, FLOPs of 4.882 G, and Params of 8.851 M under a light intensity of 84 lux.
(2) A customized FPGA accelerator for the ECSMv3_YOLO network was designed. It involves pruning and quantization of network parameters, parallel loop unfolding of convolutions, and Winograd+activation+pooling engines, and GEMM +reordering engines further accelerate the FPGA computation of network convolutions. Utilizing dual-system memory, the FPGA achieves a high throughput of 15.56 GOP/s with a power consumption of 3.2 w, and the total latency for processing a single image is reduced to 255 ms.
(3) An FPGA-accelerated network platform was constructed, conducting experiments on acceleration engines and model lightweighting. A comparison of CPU, GPU, and AX7Z020 hardware acceleration performance validates the advantages of the FPGA acceleration strategy proposed in this paper. Compared to the results of other scholars investigating FPGA acceleration for YOLO, our approach demonstrates superior performance across evaluation metrics such as FPS, mAP, power, and energy efficiency ratio.

The paper employs AX7Z020 hardware acceleration for ECSMv3_YOLO network, achieving commendable performance and cost-effectiveness. However, constrained by hardware resources, this study focuses solely on the parallel unfolding computation of the network's input and output feature mapping. Future research could leverage higher-resource FPGA hardware to explore experiments involving parallel unfolding of a cross-combination convolution with a four-layer loop, aiming to further enhance Throughput and Energy Efficiency Ratio. Considering the effects of dust and haze in mine environments on the quality of industrial camera video transmission, future developments could incorporate a dust-clearing device for cameras, such as a rolling cover protecting

Table 8. Comparative experimental results among different scholars.

	YOLOv3 (Yu et al., 2022)	YOLOv3-tiny (Adiono et al., 2021)	YOLOv3-tiny (Yu and Bouganis, 2020)	YOLOv4-tiny (Zhang et al., 2022)	This Paper
Hardware platform	PYNQ-Z2	Ultra96 V2	ZedBoard	ZYNQ–7020	AX7Z020
Frequency	650 MHz	250 MHz	100 MHz	–	150 MHz
Bit width (bits)	Fixed–16	Fixed–8	Fixed–16	Fixed–16	Fixed–16
Power (W)	10	4.26	3.36	2.86	3.2
Throughput (GOP/s)	–	31.5	10.45	9.24	15.56
Total delay (ms)	285	121	532	376	255
GOP/s/W	–	7.4	3.11	3.23	4.86

the lens equipped with a motion brush for real-time dust removal. Additionally, research into defogging algorithms could further ensure the high-quality acquisition of conveyor belt imagery.

Acknowledgements. This work was supported by the National Nature Science Foundation of China (No. 52274153), Institute of Environment-Friendly Materials and Occupational Health special fund, Anhui University of Science and Technology, China (No. ALW2021YF10), and Shanghai Rising-Star Plan (Yangfan Program) from the Science and Technology Commission of Shanghai Municipality, China (No. 22YF1400200).

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

Adiono T, **Putra A**, **Sutisna N**, et al. (2021) Low latency YOLOv3-tiny accelerator for low-cost FPGA using general matrix multiplication principle. *IEEE Access* **9**, 141890–141913.

Bao C, **Xie T**, **Feng W**, et al. (2020) A power-efficient optimizing framework fpga accelerator based on winograd for yolo. *IEEE Access* **8**, 94307–94317.

Błażej R, **Jurdziak L**, **Kozłowski T**, et al. (2018) The use of magnetic sensors in monitoring the condition of the core in steel cord conveyor belts–Tests of the measuring probe and the design of the DiagBelt system. *Measurement* **123**, 48–53.

Chen WH, **Hsu HJ and Lin YC** (2022) Implementation of a real-time uneven pavement detection system on FPGA platforms. In *2022 IEEE International Conference on Consumer Electronics-Taiwan.* IEEE, 587–588.

Ganesh P, **Chen Y**, **Yang Y**, et al. (2022) YOLO-ReT: Towards high accuracy real-time object detection on edge GPUs. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.* 3267–3277.

Girshick R (2015) Fast R-CNN [EB/OL]. Available at https://arxiv.org/abs/1504.08083 (accessed 15 September 2021).

Guo X, **Liu X**, **Królczyk G**, et al. (2022) Damage detection for conveyor belt surface based on conditional cycle generative adversarial network. *Sensors* **22**, 3485.

Guo X, **Liu X**, **Zhou H**, et al. (2022) Belt tear detection for coal mining conveyors. *Micromachines* **13**, 449.

He K, **Gkioxari G**, **Dollar P**, et al. (2018) Mask R-CNN [EB/OL]. Available at https://arxiv.org/abs/1703.06870 (accessed 15 September 2021).

Howard AG, **Zhu M**, **Chen B**, et al. (2017) MobileNets: Efficient convolutional neural networks for mobile vision applications [EB/OL]. Available at https://arxiv.org/abs/1704.04861 (accessed 18 September 2021).

Jaderberg M, **Simonyan K and Zisserman A** (2015) Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 28.

Ji J, **Miao C and Li X** (2020) Research on the energy-saving control strategy of a belt conveyor with variable belt speed based on the material flow rate. *Plos one* **15**, e0227992.

Koonce B (2021) EfficientNet. In Koonce B (ed), *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization.* Berkeley: Apress, 109–123.

Lavin A and Gray S. (2016) Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4013–4021.

Li X, **Huang H**, **Chen T**, et al. (2022) A hardware-efficient computing engine for FPGA-based deep convolutional neural network accelerator. *Microelectronics Journal* **128**, 105547.

Li W, **Li C and Yan F** (2021) Research on belt tear detection algorithm based on multiple sets of laser line assistance. *Measurement* **174**, 109047.

Li S, **Wang Q**, **Jiang J**, et al. (2022) An efficient CNN accelerator using inter-frame data reuse of videos on FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **30**(11), 1587–1600.

Liu W, **Anguelov D**, **Erhan D**, et al. (2016) SSD: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV 2016*: 14th European Conference, 11–14 October, Amsterdam, The Netherlands, Part I 14.* Springer International Publishing, 21–37.

Liu Z, **Li J**, **Shen Z**, et al. (2017) Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision.* 2736–2744.

Liu M, **Zhu Q**, **Yin Y**, et al. (2022) Damage Detection Method of Mining Conveyor Belt Based on Deep Learning. *IEEE Sensors Journal* **22**, 10870–10879.

Lu L, **Liang Y**, **Xiao Q**, et al. (2017) Evaluating fast algorithms for convolutional neural networks on FPGAs. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).* IEEE, 101–108.

Ma Y, **Cao Y**, **Vrudhula S**, et al. (2018) Optimizing the convolution operation to accelerate deep neural networks on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **26**(7), 1354–1367.

Nguyen DT, **Nguyen TN**, **Kim H**, et al. (2019) A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **27**(8), 1861–1873.

Qiu J, **Wang J**, **Yao S**, et al. (2016) Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the* 2016 *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.* 26–35.

Qu D, **Qiao T**, **Pang Y**, et al. (2020) Research on ADCN method for damage detection of mining conveyor belt. *IEEE Sensors Journal* **21**, 8662–8669.

Redmon J, **Divvala S**, **Girshick R**, et al. (2016) You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 779–788.

Ribeiro RG, **Júnior JRC**, **Cota LP**, et al. (2019) Unmanned aerial vehicle location routing problem with charging stations for belt conveyor inspection system in the mining industry. *IEEE Transactions on Intelligent Transportation Systems* **21**, 4186–4195.

Salim O, **Dey S**, **Masoumi H**, et al. (2021) Crack monitoring system for soft rock mining conveyor belt using UHF RFID sensors. *IEEE Transactions on Instrumentation and Measurement* **70**, 1–12.

Terven J and Cordova-Esparza D (2023) A comprehensive review of YOLO: From YOLOv1 and beyond. arXiv preprint arXiv:2304.00501.

Trybała P, **Blachowski J**, **Błażej R**, et al. (2020) Damage detection based on 3d point cloud data processing from laser scanning of conveyor belt surface. *Remote Sensing* **13**, 55.

Wang Y, **Wang Y**, **Dang L** (2020) Video detection of foreign objects on the surface of belt conveyor underground coal mine based on improved SSD. *Journal of Ambient Intelligence and Humanized Computing* 1–10.

Wang Q, **Wu B**, **Zhu P**, et al. (2020) ECA-Net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 11534–11542.

Xianguo L, **Lifang S**, **Zixu M**, et al. (2018) Laser-based on-line machine vision detection for longitudinal rip of conveyor belt. *Optik* **168**, 360–369.

Xu S, **Zhou Y**, **Huang Y**, et al. (2022) YOLOv4-tiny-based coal gangue image recognition and FPGA implementation. *Micromachines* **13**(11), 1983.

Yang R, **Qiao T**, **Pang Y**, et al. (2020) Infrared spectrum analysis method for detection and early warning of longitudinal tear of mine conveyor belt. *Measurement* **165**, 107856.

Yu Z and Bouganis CS (2020) A parameterisable FPGA-tailored architecture for YOLOv3-tiny. In Rincón F, Barba J, So H, Diniz P, and Caba J (eds), *Applied Reconfigurable Computing. Architectures, Tools, and Applications: 16th International Symposium, ARC 2020.* Cham: Springer International Publishing, 330–344.

Yu L, **Zhu J**, **Zhao Q**, et al. (2022) An efficient YOLO algorithm with an attention mechanism for vision-based defect inspection deployed on FPGA. *Micromachines* **13**(7), 1058.

Zhang M, **Cao Y**, **Jiang K**, et al. (2022) Proactive measures to prevent conveyor belt failures: Deep learning-based faster foreign object detection. *Engineering Failure Analysis* **141**, 106653.

Zhang F, **Li Y and Ye Z** (2022) Apply Yolov4-tiny on an FPGA-based accelerator of convolutional neural network for object detection. *Journal of Physics: Conference Series* 2303, 012032.

Zhang M, **Shi H**, **Zhang Y**, et al. (2021) Deep learning-based damage detection of mining conveyor belt. *Measurement* **175**, 109130.

Zhang M, **Zhang Y**, **Zhou M**, et al. (2021) Application of lightweight convolutional neural network for damage detection of conveyor belt. *Applied sciences* **11**, 7282.