*Journal of Network Intelligence*
**Taiwan Ubiquitous Information**
 ISSN 2414-8105
Volume **10**, Number **1**, February **2025**

# FPGA Image Edge Detection Based on Sobel Operator and Convolutional Neural Network

Kang-Le Zhou*

Nanchang Institute of Technology, Nanchang 330044, P. R. China
zhoukangle2023@163.com

Xian Mu

Nanchang Institute of Technology, Nanchang 330044, P. R. China
muxian@nut.edu.cn

Bai-Qing He

University of Campania Luigi Vanvitelli, Naples 80138, Italy
Hebaiqing123@126.com

*Corresponding author: Kang-Le Zho
Received March 6, 2024, revised May 30, 2024, accepted July 7, 2024.

Abstract. *The edge localisation of the traditional Sobel algorithm is not precise enough, and the high and low thresholds are selected by manual experience, which lacks adaptivity. Given deep learning's rapid progress, attempts have been made to use convolutional neural networks to improve edge detection. However, a significant quantity of labeled data is needed for training convolutional neural networks used in edge detection, leading to high learning costs and time consumption. Therefore, an image edge detection method based on the Sobel operator and convolutional neural network is proposed to achieve low-cost and low-power edge detection by taking advantage of FPGA hardware parallel acceleration. Firstly, an 8-direction template is used to perform the convolution operation on the image to be measured and synthesise the gradient magnitude of the centre pixel to obtain more complete rough edge information. Taking the mean value as the adaptive threshold for edge detection, the adaptive change of the threshold value with the change of grey level is achieved. Then, based on the rough edge information, a refined edge detection model based on VGGNet is proposed, which mainly consists of two major parts: the feature extraction network and the lateral edge output network. The feature extraction network consists of a fully convolutional backbone network and an attention module. The side edge output network is responsible for outputting multiple edge images in a hierarchical manner, and the final output is the fusion of different edge images. Finally, the network is further guided to output refined edges by introducing Dice loss in the loss function. The effectiveness of the proposed method is verified using the ZYNQ7000-FPGA development board from Xilinx as the hardware platform. The results show that compared with mainstream edge detection algorithms, the proposed method excels in both accuracy metrics and reaches the optimum in processing speed.*
**Keywords:** FPGA; Edge detection; Convolutional neural network; Sobel operator; VGGNet; Attention module

1. **Introduction.** With the continuous development of machine vision and intelligent computers, image edge detection technology has been widely used in computer vision [1], image processing [2], robot vision [3] and other fields. Image edge detection is a basic and important task, whose role is to automatically find and locate the contour lines of

objects in an image [4], which can provide information about the shape, size, position and orientation of the objects in an image, and is one of the important steps used for object recognition [5], tracking [6], segmentation [7] and reconstruction [8].

However, in practical applications, real-time or near real-time image edge detection is required, especially in fields such as robot vision, where higher speed and real-time performance of edge detection is required. In traditional CPU- or GPU-based implementations, edge detection algorithms usually require a large amount of computing and storage resources, leading to problems such as slower computation speed, longer response time, and higher power consumption. In order to solve these problems, image edge detection based on Field Programmable Gate Array (FPGA) has gradually become a research hotspot. The FPGA has the characteristics of high parallelism, low power consumption, low latency, and high programmability, which can provide a high-speed and low-consumption computational platform for image edge detection [9]. Due to the programmability of FPGA, the edge detection algorithm can be optimised or modified according to the actual application requirements, so as to achieve the advantages of real-time and reconfigurability. By integrating multiple processors, memories, communication interfaces and other electronic components into a single FPGA chip, fast and efficient parallel computation of multiple edge detection algorithms can be realised to improve system performance and reliability. The FPGA chip can be used to flexibly select different edge detection algorithms according to the actual needs and realise diversified and multi-level image processing, which is suitable for different fields and applications.

1.1. **Related work.** FPGA-based image edge detection is a field rich with innovative solutions geared towards real-time and resource-efficient applications. The literature shows a clear direction towards the customization and optimization of algorithms to better fit the FPGA platform [10, 11]. However, the trade-offs between robustness, speed, complexity, and adaptability continue to drive research in this exciting area. Further advancements in FPGA technology and edge detection algorithms will likely provide solutions to these present-day challenges.

Singh et al. [12] proposes a parallel architecture for real-time operator-based color picture edge detection utilizing FPGA. There is a unique edge calculation for every color component in RGB space. A potential issue to explore might relate to the scalability of their solution to different image sizes and the processing speed when handling high-resolution images. Possa et al. [13] discusses an innovative multi-resolution FPGA-based architecture that is compatible with the Harris corner detector and Canny edge detector methods. One challenge that may be faced here is the balance between hardware complexity and the ability to process images at varying resolutions efficiently. Khidhir and Abdullah [14] propose a Sobel operator-based edge detection algorithm implemented on FPGA, which moves a convolution kernel across an image to detect edges. Potential difficulties could lie in the implementation details, such as optimizing resource utilization on the FPGA or improving noise resistance. Guo et al. [15] study explores a FPGA implementation of edge detection to make edge detection more accurate and less sensitive to noise. One problem that might arise here is how well the FPGA implementation can keep up with real-time requirements for video or high-speed imaging applications. Xu et al. [16] provides a design for a Sobel filter-based edge detection on a FPGA board. Issues to consider might include power efficiency considerations and the trade-offs necessary for achieving high speed without compromising detection accuracy. Taslimi et al. [17] introduces an adaptive edge detection technique for digital images on FPGA, aiming to overcome limitations of traditional edge detection methods. The adaptability of this technique to various image types and conditions could be a focal point for identifying

limitations or required improvements. Heidler et al. [18] presents the HED-UNet, which integrates edge detection with segmentation tasks for better monitoring of the Antarctic coastline.

1.2. **Motivation and contribution.** The traditional Sobel operator image edge detection in FPGA platforms lacks adaptivity and cannot change the high and low thresholds in real time according to the changes in the external environment. The edge detection based on convolutional neural network requires a large amount of annotated data and may have problems such as edge blurring, over-extraction or under-extraction. Therefore, it is not possible to obtain refined detection results on FPGA platforms, resulting in inaccurate edge pixel localisation. The main innovations and contributions of this work include:

(1) The edge detection based on convolutional neural network is divided into two stages: rough edge detection and fine edge detection.

(2) The Sobel edge detection method was improved. A rough edge detection algorithm based on adaptive Sobel operator is proposed, and the real-time detection of image information is realized by using the parallel processing characteristics of FPGA.

(3) The edge detection method based on convolutional neural network was improved. A refined edge detection model based on VGGNet is proposed, including feature extraction network and side output network. The feature extraction network is composed of the full convolution backbone network VGG-16 and the channel attention module SENet. Each level of lateral output network will calculate the loss, which, together with the final weighted fusion output, constitutes the loss of the whole network. The loss of side output network and the loss of weighted output adopt different weights, which is convenient for guiding the network to train more accurately.

## 2. Rough edge detection based on adaptive Sobel operator.

2.1. **Image positioning.** To be tested image localisation is to locate the exact position of the image to be tested in an image containing a complex background, and to intercept the part of the image containing only the image to be tested, which is more beneficial for the subsequent recognition of the characters of the image to be tested. In this work, the first step is to locate the image to be tested by colour features in FPGA. Generally, the captured image data is a colour image in RGB format, which is converted to YCbCr format with the following transformation method:

$$\begin{cases} Y = 0.257 * R + 0.504 * G + 0.098 * B + 16 \\ Cb = -0.148 * R - 0.219 * G + 0.439 * B + 128 \\ Cr = 0.439 * R - 0.368 * G - 0.071 * B + 128 \end{cases} \tag{1}$$

Due to the complexity of floating-point and multiplication operations in FPGAs, Equation (1) needs to be fixed-pointed before conversion processing is done, as shown below:

$$\begin{cases} Y = (77 * R + 150 * G + 29 * B) \gg 8 \\ Cb = (-43 * R - 85 * G + 128 * B) \gg 8 + 128 \\ Cr = (128 * R - 107 * G - 21 * B) \gg 8 + 128 \end{cases} \tag{2}$$

After extracting the image to be measured the image area to be measured is separated from the background area based on the grey level difference, i.e. binarisation. By selecting different thresholds for binarisation, the location of the image to be measured is roughly located.

2.2. **Improved Sobel operator.** The Sobel algorithm is to use the template operator in the horizontal and vertical directions to convolve with the image to be tested [19, 20], calculate the magnitude in the horizontal and vertical directions. According to the pre-given threshold for the image to be tested segmentation, in order to obtain the edge information of the image to be tested, if the function of the image to be tested is $f(x, y)$, then its gradient can be expressed as follows.

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^T \tag{3}$$

where $G_x$ and $G_y$ denote the horizontal and vertical gradients respectively. The direction of the gradient points to the direction of the maximum rate of change of the function. The direction angles and magnitudes of the vectors are shown below [21]:

$$\begin{cases} \delta(x,y) = \arctan\left(\frac{G_y}{G_x}\right) \\ f(x,y) = \mathrm{mag}(f) = \sqrt{G_x^2 + G_y^2} \end{cases} \tag{4}$$

The traditional Sobel operator convolution template has only two directions, vertical and horizontal, and the image is differentiated after weighted averaging, and it has a specific noise suppression function [22]. The performance of edge detection in the horizontal and vertical directions is much better than that in the other directions, and although the detection effect is good and the localisation accuracy is relatively high, the algorithm also has some drawbacks [23, 24]: (1) the gradient size and gradient change in both directions are only determined by utilizing the difference between adjacent pixels in the horizontal and vertical directions. Additionally, the gradient is only affected by changes in the grey gradient in both directions; (2) there are fewer templates for the gradient calculation. If the threshold of the operator is not set reasonably, it will lead to discontinuity of the detected edges; (3) the operator is weighted first and then averaged, which may lose some details of the edges, and the detected edges are coarser.

After the above analysis, since the image to be tested tastes with diverse edge directions. In order to get a full range of responses, this paper adds six different directions of templates to the traditional Sobel operator. The horizontal and vertical directions of the traditional Sobel operator are 0° and 90°, respectively. The corresponding horizontal and vertical direction operators of the target image are shown in Figure 1.

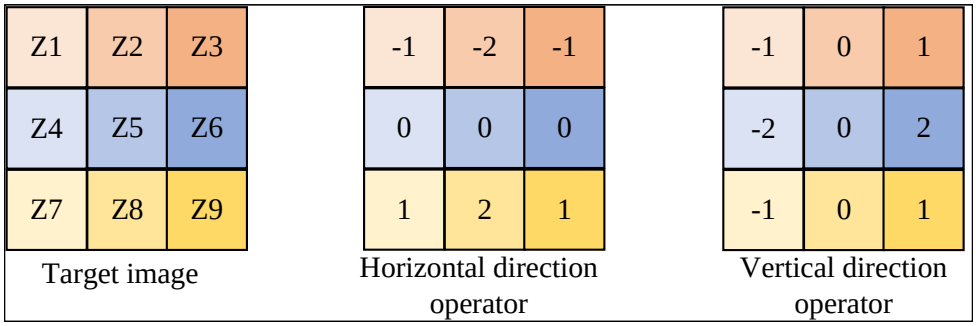


Figure 1. Traditional Sobel operator model

$$\begin{cases} G_x = (Z7 + 2Z8 + Z9) - (Z1 + 2Z2 + Z3) \\ G_y = (Z3 + 2Z6 + Z9) - (Z1 + 2Z4 + Z7) \end{cases} \tag{5}$$

The improved Sobel edge detection algorithm uses an 8-direction template to perform a convolution operation on the image image to be tested and finally synthesises the gradient

magnitude of the centre pixel, i.e. the improved algorithm detects the 8-direction gradient and introduces more templates for gradient computation to obtain more complete edge information. In order to obtain the edges of the image image to be tested, an appropriate threshold is selected to determine whether the pixel is an edge point or not. The extracted edges are displayed using a binary image with the grey value set to 1 and the screen is white for an edge point of the image; otherwise the grey value is set to 0 and the screen is black, it is not considered as an edge point.

In this paper, the traditional Sobel operator is rotated sequentially and incrementally at 45° intervals to obtain increased templates in the 45°, 90°, 135°, 180°, 225°, 270°, and 315° directions, respectively. The increased operator convolution factors are shown in Figure 2.

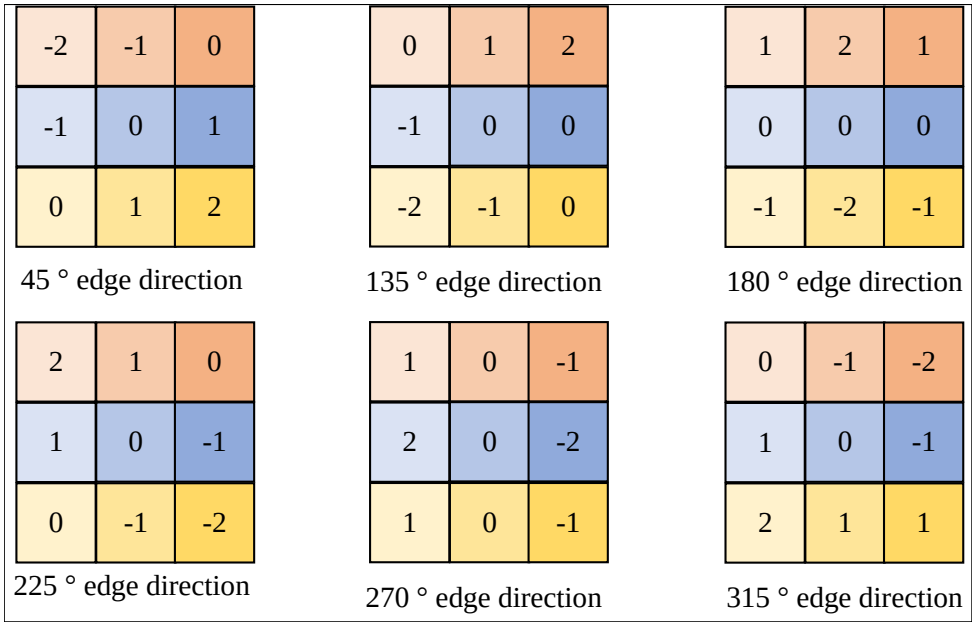


Figure 2. Improved Sobel operator model

Since the improved algorithm introduces more templates for gradient calculation, more comprehensive edge information can be obtained. The gradient calculation method can be represented by Equation (6).

$$\begin{cases} G_{45^\circ} = (Z6 + Z8 + 2Z9) - (Z1 + Z2 + Z4) \\ G_{135^\circ} = (Z2 + 2Z3 + Z6) - (Z4 + Z7 + Z8) \\ G_{180^\circ} = (Z1 + 2Z2 + Z3) - (Z7 + 2Z8 + Z9) \\ G_{225^\circ} = (Z1 + Z2 + Z4) - (Z6 + Z7 + 2Z8) \\ G_{270^\circ} = (Z1 + 2Z4 + Z7) - (Z3 + 2Z6 + Z9) \\ G_{315^\circ} = (Z4 + 2Z7 + Z8) - (Z2 + 2Z3 + Z6) \end{cases} \tag{6}$$

2.3. **Adaptive thresholding algorithm analysis.** Due to the influence of many external factors such as lights, sunlight and even staff walking in the detection process, will cause some interference to the grey scale of the detected target, and it is very likely that part of the data of the image to be detected will be lost. In order to be able to solve the impact of external factors, the edge detection threshold must change with the change of external factors. In this paper, a frame image is segmented into a number of pixel windows of $3 \times 3$ matrices, and the mean value is used as the threshold for edge detection to

achieve that the threshold changes with the change of grey level. The steps to implement adaptive thresholding in FPGA environment are as follows:

(1) Delay by two line caches to get a sliding $3 \times 3$ window template.

(2) Each row is delayed by three D triggers, which allows nine data in a $3 \times 3$ sliding template to be output simultaneously.

(3) The summing operation is performed on the nine data obtained by means of an adder.

(4) The adaptive threshold is obtained by averaging the summation results using a divider.

In order to maximally remove the noise and preserve the original boundary, dual thresholding is used to judge whether it is an edge or not. The method of double thresholding to judge the edge of an image can be expressed by Equation (7).

$$F(x,y) = \begin{cases} 1 & M \geq AGV \\ 1 \text{ or } 0 & \frac{AGV}{2} < M < AGV \\ 0 & M \leq \frac{AGV}{2} \end{cases} \tag{7}$$

where $AGV$ denotes the adaptive threshold and $M$ denotes the gradient magnitude of the centroid pixel. When the gradient magnitude of the centre pixel is greater than the adaptive threshold, then output 1. When the gradient magnitude of the centre pixel is less than half of the adaptive threshold, then output 0. When the gradient magnitude of the centroid pixel is between half of the adaptive threshold and the adaptive threshold, then it is necessary to see whether the previous centroid pixel point is an edge or not, and if it is, then output 1, and conversely, output 0.

## 3. Refined edge detection model based on VGGNet.

### 3.1. Overview of the network model.

The designed network structure is mainly based on convolutional neural networks and attention mechanisms, and consists of two main parts: the feature extraction network and the side output network. The overall network adopts a similar structure to HED, but with the following different improvements:

(1) A channel attention module was introduced in the fully convolutional backbone network (VGG-16) [25] to enhance feature extraction.

(2) HED proposed a new loss function to alleviate the problem of positive and negative sample imbalance, which is based on the cross-entropy loss function, and although better results were obtained, the loss function constructed only from the pixel-level perspective can not accurately locate the edge pixels, and the final results obtained are very rough and not fine enough. To address this problem, this paper constructs a fusion loss function by introducing an image-level loss function that is fused with the cross-entropy loss function, effectively alleviating the problem of too rough output edges.

### 3.2. Feature extraction network.

A Fully Convolutional Network (FCN) [26] is used as the backbone network. However, unlike FCN which converts fully connected layers into convolutional layers, the backbone network in this paper is based on the VGG-16 network model and removes all fully connected layers and the last pooling layer, which is shown in Figure 3.

The choice of VGG-16 as the backbone network is mainly based on the following two considerations: (1) The backbone network should be very deep, but in order to ensure real-time performance, the computational load and parameter count should be as small as possible. VGGNet successfully constructs a deep convolutional neural network of 16 to 19 layers and greatly reduces the number of parameters by iteratively stacking the $3 \times 3$

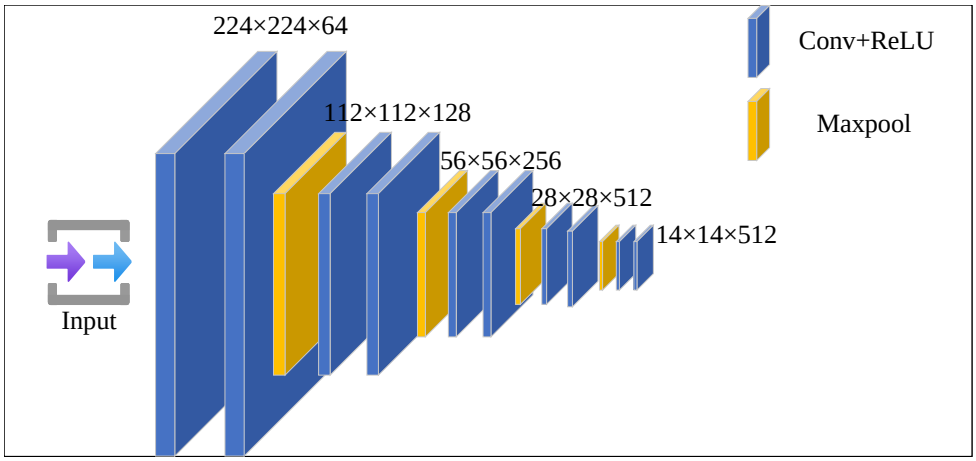


Figure 3. VGG-16 based backbone network structure

convolutional kernel and the $2 \times 2$ pooling layer. (2) The backbone network should have multiple scales. In this paper, we want to construct an end-to-end model and make full use of the feature maps of each stage as a way to improve the final detection accuracy. The VGG-16 is characterised by an architecture divided into five stages, each consisting of a downsampling layer with a step size of 2 (the pooling layer), and each stage possesses a different scale of features.

In order to enhance the feature extraction capability of a fully convolutional backbone network, a channel attention module is introduced into it. In convolutional neural networks, the attention module is an extra neural network that may choose certain sections of the input or allocate varying weights to various sections of the input. This leads to enhanced network performance and reduced computational power use. The introduced channel attention module is SENet (Squeeze-and-Excitation Net) [27, 28], as shown in Figure 4.

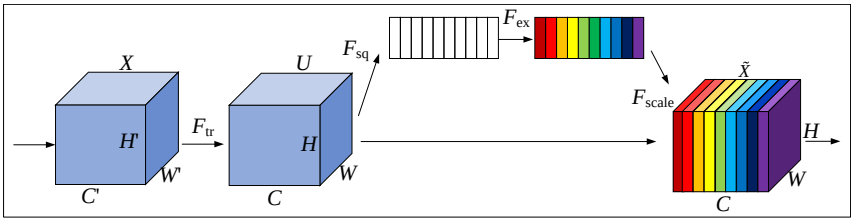


Figure 4. SENet channel attention module

The central idea of SENet is to guide the network to update the weights according to the loss function, so that the feature maps with good results have more weights and the feature maps with ineffective or poor results have less weights, so as to achieve better results. SENet is divided into two operations: Squeeze and Excitation.

Squeeze: Suppose the input image $X$ is convolved with the feature vector $U = [u_1, u_2, u_3, \ldots, u_c]$. The size of the original feature map $U$ is compressed from $H \times W \times C$ to $1 \times 1 \times C$ by using global average pooling [29].

$$Z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \tag{8}$$

where $H$ and $W$ represent the height and width of the input image, respectively.

Excitation: The feature map $U$ is transformed into a channel descriptor $Z_c$ after a squeezing operation.

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma\left(W_2 \delta\left(W_1 z\right)\right) \tag{9}$$

where $\delta$ denotes the ReLU activation function, $\sigma$ denotes the Sigmoid activation function. $W_1$ and $W_2$ denote the weight parameters of the two fully connected layers respectively. Experimentally, it is proved that the optimal result is obtained when narrowing down the parameter $r = 16$, so $r$ is set to 16 here.

The final output of SENet is shown below:

$$X_F = F_{scale}\left(u_c, s_c\right) = s_c u_c \tag{10}$$

The structure of SENet is simple and the main increase in parameters throughout the module comes from the two fully connected layers, so the increase can be calculated from Equation (10).

$$\Delta Param = \frac{2}{r} \sum_{s=1}^{S} N_s C_s^2 \tag{11}$$

where $N_s$ denotes the number of SE modules used, and $C_s$ denotes the number of channels in the excitation operation. The increase in the number of parameters is very small, but it can greatly improve the feature extraction capability of the backbone network.

Based on the above analysis of parametric quantity increment, this paper introduces SENet after each convolutional layer of the backbone network to fully enhance the feature extraction capability of the backbone network. The full convolutional backbone network and the channel attention module together form the feature extraction network, which makes the feature extraction network output more effective feature maps.

3.3. **Side output networks.** Losses are computed for each level of the sideloop output network and, together with the final weighted fused output, form the loss of the entire network. Different weights are used for the loss of the side output network and the loss of the weighted output to facilitate more accurate guidance of the network for training. Assuming that the whole network has $M$ side output networks, each of which is connected to a classifier, where the corresponding weights are denoted as $w = \left(w^{(1)}, \ldots, w^{(M)}\right)$, then the objective function can be defined as:

$$L_{side}\left(W, w\right) = \sum_{m=1}^{M} l_{side}^{m}\left(W, w^{(m)}\right) \tag{12}$$

where $l_{side}$ denotes the loss function in the side-by-side output network at each level [30, 31].

During the end-to-end training process, the loss function is used to guide the network by calculating the difference between all pixel points $X = (x_i, i = 1, \ldots, |X|)$ of the input image and the corresponding labelled image $Y = (y_i, i = 1, \ldots, |Y|)$.

In order to utilise the outputs of each level of the side output network, all the side outputs are weighted and fused, and the difference between the fused results and the labelled images is also computed into the loss, and this part of the loss function can be defined as:

$$L_{fuse}\left(W, w, h\right) = \text{Dist}\left(Y, P_{fuse}\right) \tag{13}$$

where $P_{fuse}$ denotes the result after weighted fusion, $h = (h_1, \ldots, h_M)$ denotes the fusion weight of each layer, and $\mathrm{Dist}(\cdot)$ denotes the difference between the fused output and the binary labelled image [32].

Therefore, the backpropagation process of the whole network consists of the above two parts (Equation (11) and Equation (12)). Putting the two parts together constitutes the backpropagation of the whole network as follows.

$$L(W, w, h) = \arg\min \left(L_{side}(W, w), L_{fuse}(W, w, h)\right) \tag{14}$$

During testing, the input image $X$, the side output network and the final fusion weighting output edge prediction maps respectively.

$$\left(P_{fuse}, P_{side}^{(1)}, \ldots, P_{side}^{(M)}\right) = CNN\left(X, (W, w, h)^*\right) \tag{15}$$

where $CNN(\cdot)$ denotes the output edge map of the entire network [33]. The final uniform output result can be obtained by further averaging these output edge maps as follows:

$$P_{out} = \mathrm{Average}\left(P_{fuse}, P_{side}^{(1)}, \ldots, P_{side}^{(M)}\right) \tag{16}$$

3.4. **Weighted cross-entropy loss function.** Problems involving binary classification are the most common applications of the conventional cross entropy loss function, but the edge detection task is extremely unbalanced due to the extreme imbalance in the distribution of edge/non-edge pixels, so the directly applying of the cross entropy loss function [34, 35] can be a great obstacle to the training of the network.

This paper introduces the image-level loss function Dice Loss for edge detection based on the cross-entropy loss function. It is common practice to utilize the dice coefficient, an ensemble similarity measure function, to determine how similar two samples are to one another. Assuming that the input image is $X$, the corresponding labelled binary image is $Y$, and the output predicted image of the model is $P$, according to the Dice coefficient, the Dice Loss loss function can be obtained as follows.

$$L(Y, P) = \mathrm{Dist}(Y, P) = \frac{\sum_{i=1}^{N} y_i^2 + \sum_{i=1}^{N} p_i^2}{2\sum_{i=1}^{N} y_i p_i} \tag{17}$$

where $y_i$ and $p_i$ denote the $i$-th pixel value of the binary labelled image $Y$ and the model predicted output image $P$, respectively.

The purpose of Dice Loss loss function is to compare the similarity of two images, and then minimise the distance between them. Dice Loss does not need to consider the use of interclass balancing weights $\beta$ and $1 - \beta$ to alleviate the problem of interclass imbalance, so that the network can be trained at the same time, and directly output a more detailed edge image. In order to get better results, this paper makes a weighted fusion of the standard cross-first loss function and the Dice Loss loss function, such a fusion loss function can make the whole network subject to the constraints of two angles, i.e., the image level and the pixel level, and the final loss function is shown as follows:

$$L_{\text{final}}(Y, P) = \alpha L(Y, P) + \beta L_c(Y, P) \tag{18}$$

where $L_c(\cdot)$ represents the standard cross-first loss function; $\alpha$ and $\beta$ represent the weights of the two loss functions.

In summary, the improved loss function, with image-level constraints, no longer only considers the problem from the perspective of pixel level, so that the network can output a finer edge image and locate the edge pixels more accurately.

## 4. Experimental results and analyses.

**4.1. Experimental dataset and experimental environment.** In the field of edge detection, the commonly used public datasets are BSDS500, PASCAL VOC, NYUD v2, etc. These public datasets have different characteristics, among which the most commonly used public dataset is BSDS500, which contains 200 training images, 100 validation images, and 200 test images, and all the labelled images are saved in .mat files, and each labelled image has 5 manually labelled truth values for each labelled image. The experiments are carried out on Vivado 2019 development environment and the FPGA master clock is 50 MHz. The code is written using Verilog HDL and simultaneously compiled as well as downloaded and burned into the development board for image acquisition, adaptive Sobel rough edge detection and VGGNet refined edge detection.

The experimental device system is based on the XILINX-ZYNQ7000-FPGA platform implementation, the platform and the traditional embedded CPU, compared with a strong parallel processing capability, can meet the different algorithms module to achieve the demand for hardware acceleration, while the integration of the ARM architecture, both can be complex algorithms of process control and parallel computing advantages.

The whole FPGA+ARM-based architecture can be divided into programmable logic part and processing system part. In order to reduce the training time, make the model converge faster, and improve the detection accuracy, the proposed backbone network VGG-16 in this paper is first pre-trained on the large-scale public dataset ImageNet. After obtaining the pre-training weights, the Adam optimisation algorithm is used to update the network weights. During training, the number of batches is set to 22, the weight decay is 0.0005, and the base learning rate is set to 0.0001.

**4.2. Rough edge detection results.** The rough edge detection method based on adaptive Sobel operator is firstly validated. The qualitative comparison results of the proposed adaptive Sobel operator with the traditional Sobel operator are shown in Figure 5.

(a) Original image (b) Traditional Sobel (c) Adaptive Sobel

Figure 5. Qualitative Comparison of Representative Test Results

It can be seen that the adaptive Sobel edge detection using eight directions can locate the edge points more accurately and the image information is more complete, and the edge image is obviously more three-dimensional. In the absence of light interference, the

improved adaptive threshold can reflect the edge information of the image to a large extent, so it is also more resistant to light interference.

4.3. **Comparison of results with mainstream algorithms.** In order to verify the effectiveness of the proposed method, some mainstream edge detection algorithms are selected for comparison experiments on the BSDS500 dataset, including Canny operator, Sobel operator, CED, HED and other mainstream edge detection algorithms.

To be fair, the backbone networks of CED and HED are both VGGNet. The comparison results are shown in Table 1. Among them, ODS (Optimal Dataset Scale), OIS (Optimal Image Scale) and FPS (Frames Per Second) are the commonly used evaluation metrics for edge detection algorithms. ODS is a comprehensive performance evaluation metric. OIS is able to reflect the comprehensive situation of the model's performance on different images. FPS refers to the number of frames that the algorithm FPS is the number of image frames processed per second.

Table 1. Quantitative comparison of different edge detection algorithms

| **Algorithm** | **ODS** | **OIS** | **FPS** |
|---|---|---|---|
| Canny | 0.603 | 0.659 | 23 |
| Sobel | 0.646 | 0.671 | 25 |
| CED | 0.749 | 0.812 | 30 |
| HED | 0.773 | 0.806 | 25 |
| Ours | 0.786 | 0.806 | 30 |

It can be seen that the Ours algorithm has the highest score on the ODS at 0.786, which means that the Ours algorithm is able to achieve the highest edge detection accuracy among all the algorithms considered with optimal fixed thresholds on the entire dataset. On the OIS score, the Ours algorithm and the CED algorithm are tied for the highest score at 0.806. This shows that the Ours and CED algorithms can achieve the best average performance on the dataset when the optimal thresholds are selected individually for each image, respectively. This again shows that the Ours algorithm has good consistency and effectiveness in processing different images. For FPS, the Ours algorithm is the same as the CED algorithm at 30. This is the highest value in the table and indicates that the Ours algorithm is not only highly accurate but also fast enough to process up to 30 frames per second. This is especially important for applications that require real-time edge detection.

To summarise, the Ours algorithm performs well on both the accuracy metrics ODS and OIS, and optimally on the processing speed FPS. This performance makes the Ours algorithm a strong candidate for edge detection, especially when fast and accurate edge detection is required. The remaining algorithms, such as CED, also perform well on OIS and HED ranks second on ODS, but Ours has the combined advantage of high accuracy and speed.

5. **Conclusion.** This work proposes to divide edge detection based on convolutional neural networks into two stages: rough edge detection and refined edge detection. Firstly, a rough edge detection algorithm based on adaptive Sobel operator is proposed to achieve real-time detection of image information by using the characteristics of FPGA parallel processing. Then, an edge detection method based on convolutional neural network is improved. A refined edge detection model based on VGGNet is proposed, including a feature extraction network as well as a side edge output network. The feature extraction network consists of the full convolutional backbone network VGG-16 and the channel

attention module SENet. Losses are computed for each level of the side edge output network, and together with the final weighted fused output, form the loss of the entire network. Different weights are used for the loss of the sidelobe output network and the loss of the weighted output, which facilitates more accurate guidance of the network for training. The proposed algorithm performs well in both accuracy metrics ODS and OIS, and optimally in processing speed FPS, resulting in fast and accurate FPGA image edge detection.

**Acknowledgment.** This work is supported by the Science and Technology Project of Jiangxi Provincial Department of Education (No. GJJ2202711, No. GJJ2202722).

## REFERENCES

[1] J. Jing, S. Liu, G. Wang, W. Zhang, and C. Sun, "Recent advances on image edge detection: A comprehensive review," Neurocomputing, vol. 12, no. 3, 147, 2022.
[2] R. Muthukrishnan, and M. Radha, "Edge detection techniques for image segmentation," International Journal of Computer Science & Information Technology, vol. 3, no. 6, 259, 2011.
[3] G. Shrivakshan, and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," International Journal of Computer Science Issues (IJCSI), vol. 9, no. 5, 269, 2012.
[4] E. Nadernejad, S. Sharifzadeh, and H. Hassanpour, "Edge detection techniques: Evaluations and comparisons," Applied Mathematical Sciences, vol. 2, no. 31, pp. 1507-1520, 2008.
[5] M. Kumar, and R. Saxena, "Algorithm and technique on various edge detection: A survey," Signal & Image Processing, vol. 4, no. 3, 65, 2013.
[6] T.-Y. Wu, H. Li, S. Kumari, and C.-M. Chen, "A Spectral Convolutional Neural Network Model Based on Adaptive Fick's Law for Hyperspectral Image Classification," Computers, Materials & Continua, vol. 79, no. 1, pp. 19-46, 2024.
[7] F. Zhang, T.-Y. Wu, J.-S. Pan, G. Ding, and Z. Li, "Human motion recognition based on SVM in VR art media interaction environment," Human-centric Computing and Information Sciences, vol. 9, 40, 2019.
[8] F. Zhang, T.-Y. Wu, and G. Zheng, "Video salient region detection model based on wavelet transform and feature comparison," EURASIP Journal on Image and Video Processing, vol. 2019, 58, 2019.
[9] M. Mittal, A. Verma, I. Kaur, B. Kaur, M. Sharma, L. M. Goyal, S. Roy, and T.-H. Kim, "An efficient edge detection approach to provide better edge connectivity for image analysis," IEEE Access, vol. 7, pp. 33240-33255, 2019.
[10] H. S. Neoh, and A. Hazanchuk, "Adaptive edge detection for real-time video processing using FPGAs," Global Signal Processing, vol. 7, no. 3, pp. 2-3, 2004.
[11] T. Abbasi, and M. Abbasi, "A novel FPGA-based architecture for Sobel edge detection operator," International Journal of Electronics, vol. 94, no. 9, pp. 889-896, 2007.
[12] S. Singh, A. K. Saini, and R. Saini, "Real-time FPGA based implementation of color image edge detection," International Journal of Image, Graphics and Signal Processing, vol. 4, no. 12, 19, 2012.
[13] P. R. Possa, S. A. Mahmoudi, N. Harb, C. Valderrama, and P. Manneback, "A multi-resolution FPGA-based architecture for real-time edge and corner detection," IEEE Transactions on Computers, vol. 63, no. 10, pp. 2376-2388, 2013.
[14] A. M. Khidhir, and N. Y. Abdullah, "FPGA based edge detection using modified sobel filter," International Journal for Research and Development in Engineering, vol. 2, no. 1, pp. 22-32, 2013.
[15] Z. Guo, W. Xu, and Z. Chai, "Image edge detection based on FPGA." pp. 169-171.
[16] Q. Xu, S. Varadarajan, C. Chakrabarti, and L. J. Karam, "A distributed canny edge detector: algorithm and FPGA implementation," IEEE Transactions on Image Processing, vol. 23, no. 7, pp. 2944-2960, 2014.
[17] S. Taslimi, R. Faraji, A. Aghasi, and H. R. Naji, "Adaptive edge detection technique implemented on FPGA," Iranian Journal of Science and Technology, Transactions of Electrical Engineering, vol. 44, pp. 1571-1582, 2020.
[18] K. Heidler, L. Mou, C. Baumhoer, A. Dietz, and X. X. Zhu, "HED-UNet: Combined segmentation and edge detection for monitoring the Antarctic coastline," IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-14, 2021.
[19] J. J. Heckman, "Rejoinder: response to Sobel," Sociological Methodology, vol. 35, no. 1, pp. 135-150, 2005.

[20] G. Chen, Z. Jiang, and M. Kamruzzaman, "Radar remote sensing image retrieval algorithm based on improved Sobel operator," Journal of Visual Communication and Image Representation, vol. 71, 102720, 2020.
[21] R. Chetia, S. Boruah, and P. Sahu, "Quantum image edge detection using improved Sobel mask based on NEQR," Quantum Information Processing, vol. 20, pp. 1-25, 2021.
[22] R. Tian, G. Sun, X. Liu, and B. Zheng, "Sobel edge detection based on weighted nuclear norm minimization image denoising," Electronics, vol. 10, no. 6, 655, 2021.
[23] Y. Ma, H. Ma, and P. Chu, "Demonstration of quantum image edge extraction enhancement through improved Sobel operator," IEEE Access, vol. 8, pp. 210277-210285, 2020.
[24] C. Sitaula, and M. B. Hossain, "Attention-based VGG-16 model for COVID-19 chest X-ray image classification," Applied Intelligence, vol. 51, pp. 2850-2863, 2021.
[25] M. Ye, N. Ruiwen, Z. Chang, G. He, H. Tianli, L. Shijun, S. Yu, Z. Tong, and G. Ying, "A lightweight model of VGG-16 for remote sensing image classification," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 6916-6922, 2021.
[26] F. Ahmed, M. Asif, M. Saleem, U. F. Mushtaq, and M. Imran, "Identification and Prediction of Brain Tumor Using VGG-16 Empowered with Explainable Artificial Intelligence," International Journal of Computational and Innovative Sciences, vol. 2, no. 2, pp. 24-33, 2023.
[27] F. Zhang, G. Ding, Z. Mao, L. Xu, and X. Zheng, "Bayesian Network for Motivation Classification in Creative Computation," Journal of Information Hiding and Multimedia Signal Processing, vol. 8, no. 4, pp. 888-902, 2017.
[28] F. Zhang, Z. Mao, G. Ding, and L. Xu, "Design of Chinese Natural Language in Fuzzy Boundary Determination Algorithm Based on Big Data," Journal of Information Hiding and Multimedia Signal Processing, vol. 8, no. 2, pp. 423-434, 2017.
[29] Y. A. Farrukh, S. Wali, I. Khan, and N. D. Bastian, "Senet-i: An approach for detecting network intrusions through serialized network traffic images," Engineering Applications of Artificial Intelligence, vol. 126, 107169, 2023.
[30] J. Huang, L. Ren, X. Zhou, and K. Yan, "An improved neural network based on SENet for sleep stage classification," IEEE Journal of Biomedical and Health Informatics, vol. 26, no. 10, pp. 4948-4956, 2022.
[31] F. Zhang, Y. Wang, P. Mei, A. Dai, B. Wang, L. Liu, and Y. Xia, "Study on Gene Splicing Site Recognition Based on Particle Swarm Optimization Twin Support Vector Machine Algorithm for Smart Healthcare," Wireless Communications and Mobile Computing, vol. 2023, 4097660, 2023.
[32] J. Xiong, X.-L. Meng, Z.-Q. Chen, C.-S. Wang, F.-Q. Zhang, A. Grau, and Y. Chen, "One-Dimensional EEG Artifact Removal Network Based on Convolutional Neural Networks," Journal of Network Intelligence, vol. 9, no. 1, pp. 142-159, 2024.
[33] Y. Huang, P. Shi, H. He, H. He, and B. Zhao, "Senet: spatial information enhancement for semantic segmentation neural networks," The Visual Computer, vol. 14, pp. 1-14, 2023.
[34] D. Cheng, G. Meng, G. Cheng, and C. Pan, "SeNet: Structured edge network for sea–land segmentation," IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 2, pp. 247-251, 2016.
[35] T.-Q. Huang, C.-S. Wang, Z.-Q. Chen, F.-Q. Zhang, X.-L. Meng, A. Grau, Y. Chen and J.-W. Huang, "2-D GCANet Applied to Denoise 1-D EEG Signals in Online Remote Teaching Scenarios," Journal of Network Intelligence, vol. 8, no. 4, pp. 1289-1302, 2023.