electronics

MDPI

Article

FOLD: Low-Level Image Enhancement for Low-Light Object Detection Based on FPGA MPSoC

Xiang Li [1,2], Zeyu Li [1,*], Lirong Zhou [3] and Zhao Huang [4,*]

[1] School of Computer Science and Technology, North University of China, Taiyuan 030051, China; p233343@hsu.edu.hk
[2] School of Decision Sciences, The Hang Seng University of Hong Kong, Hong Kong 999077, China
[3] School of Computer Science and Technology, Xidian University, Xi'an 710071, China; zlrong@stu.xidian.edu.cn
[4] Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China
* Correspondence: 20230101@nuc.edu.cn (Z.L.); z_huang@xidian.edu.cn (Z.H.)

Abstract: Object detection has a wide range of applications as the most fundamental and challenging task in computer vision. However, the image quality problems such as low brightness, low contrast, and high noise in low-light scenes cause significant degradation of object detection performance. To address this, this paper focuses on object detection algorithms in low-light scenarios, carries out exploration and research from the aspects of low-light image enhancement and object detection, and proposes low-level image enhancement for low-light object detection based on the FPGA MPSoC method. On the one hand, the low-light dataset is expanded and the YOLOv3 object detection model is trained based on the low-order image enhancement technique, which improves the detection performance of the model in low-light scenarios; on the other hand, the model is deployed on the MPSoC board to achieve an edge object detection system, which improves the detection efficiency. Finally, validation experiments are conducted on the publicly available low-light object detection dataset and the ZU3EG-AXU3EGB MPSoC board, and the results show that the method in this paper can effectively improve the detection accuracy and efficiency.

Keywords: object detection; low-light image; image enhancement; FPGA MPSoC

Citation: Li, X.; Li, Z.; Zhou, L.; Huang, Z. FOLD: Low-Level Image Enhancement for Low-Light Object Detection Based on FPGA MPSoC. *Electronics* **2024**, *13*, 230. https://doi.org/10.3390/electronics13010230

Academic Editor: Alexander Barkalov

Received: 25 November 2023
Revised: 29 December 2023
Accepted: 30 December 2023
Published: 4 January 2024

1. Introduction

Target detection, as a fundamental task and research frontier in computer vision, is an important foundation for other vision tasks such as instance segmentation [1] and target tracking [2]. However, in some complex low-light scenes, the accuracy of target detection methods is often also affected by image quality issues, and many practical applications based on target detection have encountered difficulties as a result. For example, images captured in low-light scenes often contain a lot of noise due to the light-sensitive performance of the sensor. Therefore, the problem of target detection in low-light scenes is of great practical importance.

The problem of low-light target detection can be divided into two sub-problems: low-light image enhancement and generic target detection, where low-light image enhancement is used to recover high-quality normal-light images from low-quality low-light images, which is the key link to perform downstream target detection tasks [3]. Traditional low-light image enhancement methods mainly start from basic image signal processing theories and methods, and manually design filters or grey-scale mapping methods for image filtering or light estimation to obtain high-quality normal-light images [4]. However, these methods require a large amount of a priori knowledge and expert experience for manual tuning, and it is difficult to balance multiple image quality degradation problems in complex scenes. In recent years, deep learning-based low-light image enhancement methods have achieved better enhancement results than the traditional ones, because deep neural networks can

efficiently learn the mapping from low-light to normal-light distributions through end-to-end training from paired low-light/normal-light image datasets [5]. However, paired data are very expensive to collect and synthetic data cannot be adapted to complex real-world scenarios. Therefore, how to use unpaired data for training and how to effectively improve the performance of the enhanced target detection task is a valuable research topic.

Existing general-purpose target detection algorithms work well in clear images with normal lighting, but when applied without modification to complex low-light scenes, the performance of the algorithms is severely compromised [6]. However, most approaches are still designed for human vision and focus on improving the quality of human visual perception rather than improving the performance of the target detection task. In addition, most current applications need to be implemented at the edge, where power consumption and performance are important constraints. Therefore, how to combine shimmering image enhancement with target detection algorithms and use them in MPSoC implementations is a key research topic to solve the target detection problem in shimmering scenes.

Based on the above research background, we explores the target detection algorithms in low-light scenarios, and proposes low-level image enhancement for low-light object detection based on the MPSoC method. For low-light image enhancement, we adopt three low-order image enhancement techniques to improve the image quality, and enhance the existing low-light image data and realistically collected data to expand the training dataset, which strengthens the foundation for training the downstream target detection model; for general-purpose target detection, we train the YOLOv3 target detection model based on the above dataset, and deploy it on the FPGA MPSoC to achieve high-performance detection. The method proposed in this paper organically combines low-light image enhancement and target detection models, and the overall research framework is shown in Figure 1. To this end, we make the following contributions:

1. Three low-order data augmentation techniques were used to augment ExDark and realistically acquired low-light image data to expand the training dataset.
2. The YOLOv3 target detection model based on PyTorch was deployed on FPGA MPSoC for improving target detection performance.
3. Low-light image enhancement and target detection models have been combined to solve the target detection problem in low-light scenes.

The paper is structured as follows. Section 2 discusses related work on low-light image enhancement and universal object detection. In Section 3, we introduce the dataset and the image enhancement technology. In Section 4, we provide a detailed description of the implementation of the model deployment on MPSoC. In Section 5, we evaluate the feasibility of our approach. Section 6 concludes the paper.

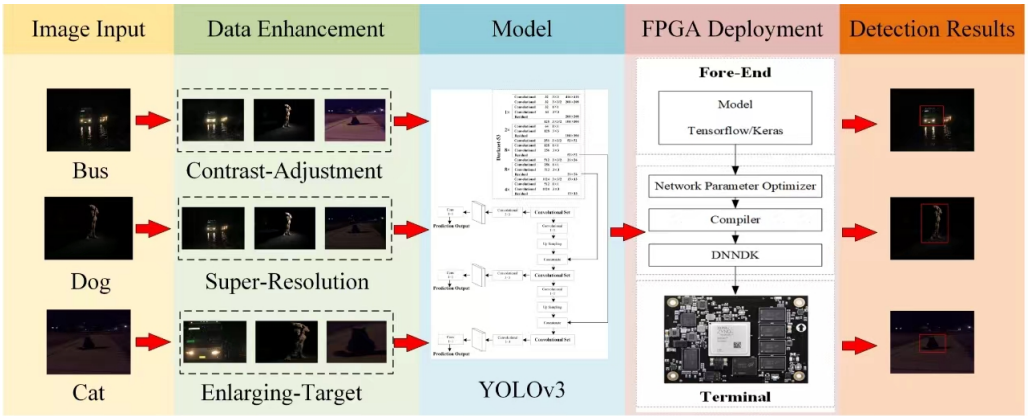

Figure 1. The overall framework of the FOLD.

2. Related Work

In this section, we briefly describe two topics relevant to our work. The first subsection describes existing methods for improving shimmering images. However, they mostly

attempt to improve perceptual quality rather than object recognition performance. The second subsection describes the studies of research on improving downstream visual tasks through image enhancement techniques, and how deploying models on FPGA-based MPSoCs can improve the performance of these techniques in edge computing scenarios.

2.1. Low-Light Image Enhancement

Actually, the quality of low-light images taken under poor lighting conditions is usually very poor. Researchers from academic and industrial sectors have attempted to recover high-quality images from low-light images through image enhancement methods. Traditional methods primarily includes two categories: histogram-based methods and retinex-based methods. The histogram-based methods, i.e., Adaptive Histogram Equalization (AHE), can map the histogram of the entire image pixel by pixel to a new distribution. The idea of retinex theory [7] is intended to separate illumination and reflectance, and retinex-based methods typically utilize illumination maps to enhance image quality. However, due to the large amount of noise in shimmering images, the performance of the conventional methods is poor.

Recently, deep learning (DL)-based approaches have performed well in many low-level vision tasks, such as denoising [8]. Lore et al. [9] presented a depth autoencoder-based solution which can simultaneously enhance the contrast of shimmering images as well as denoising them. Wei et al. [10] incorporated DL algorithms with retinex theory to propose an end-to-end framework for decomposition and illumination enhancement. The EEMEFN method in [11] applied a multi-exposure fusion module and an edge enhancement module for very low-illumination image enhancement.

Different from the approaches which aim to improve image quality for better human visual perception, this paper presents a shimmering image enhancement system for edge computing scenarios which is intended to increase the performance of downstream visual tasks. The target detection performance is utilized to quantitatively evaluate the effectiveness of our approach, rather than from a perceptual quality perspective.

2.2. Deployment of Object Detection Models at the Edge

In recent years, research has highlighted the degradation of image quality on downstream visual tasks. Similarly, several researchers have attempted to exploit image enhancement techniques to improve downstream visual tasks, such as image classification [12], action recognition [13], and object detection [14]. Despite these methods offering limited improvements in the perceptual quality of images, they can significantly increase the performance of downstream visual tasks. Costa et al. [12] considered different types of noise and noise levels and showed that these denoising methods can improve the classification accuracy of noisy data. Kvyetnyy et al. [14] proposed an image denoising method based on bilateral filtering, wavelet threshold, as well as enhancement methods for target detection. Bai et al. [15] focused on small target detection and proposed an end-to-end generative adversarial network to improve the detection performance of small targets.

In terms of hardware implementations, while GPUs have been shown to provide extremely high throughput and are widely used for hardware acceleration of DNNs, they are often not suitable for energy/power-constrained applications such as edge computing scenarios due to their high power consumption [16]. FPGAs allow us to implement irregular parallelism, custom data types, and application-specific hardware architectures, providing great flexibility to adapt to newer DNN models with higher sparsity and compact network structures [17,18].

In this paper, we focus on enhancing and extending low-light image data to strengthen the foundation for downstream vision task model training, as well as deploying the target detection model on FPGA-based MPSoCs to help the detector improve its performance to serve edge computing scenarios.

3. Training Datasets and Enhancement Technology

3.1. Low-Light Image Training Datasets

Existing large-scale genera- purpose target recognition datasets contain only a very small number of low-light image samples, e.g., Microsoft COCO [19], ImageNet [20], and PASCAL VOC [21], whose low-light images account for only about 1% of all samples, making it difficult to use them effectively for training low-light image recognition models. Therefore, the public dataset ExDark [22], which consists of all low-light images with target-level annotations, is selected as the base dataset in this paper. The ExDark dataset contains 7363 low-light images with 12 different target classes. Among them, the training set contains 3000 images with 250 images in each category, the validation set contains 1800 images with 150 images in each category, and the test set contains 2563 images. Most of the images in the dataset were downloaded from search engines, and a few images were selected from the large public datasets mentioned above (Microsoft COCO, etc.). The resolution and aspect ratios of these images vary widely, and the quality of the images captured is variable. In addition to the target types, ExDark includes information on 10 different low-light scene categories, ranging from very low light to low light for complex low-light scenes. It is worth noting that the dataset does not contain normal exposure images paired with low-light images, making it difficult to apply supervised low-light image enhancement algorithms to them.

The ExDark dataset contains a wide variety of scenes covering common low-light scene types: (a) very low-light images; (b) low-light images with no light source; (c) low-light images with an illuminated object surrounded by darkness; (d) low-light images with a single visible light source; (e) low-light images with multiple, but weaker light sources; (f) low-light images with multiple, but stronger light sources; (g) low-light images with visible screens; (h) low-light images of an interior room with brightly lit windows; (i) low-light images of an interior room with brightly lit screens; (j) low-light images of an interior room; (k) low-light images of a target to be inspected in shadow; (l) low-light images of a target to be inspected in darkness; and (m) low-light images of a target to be inspected in a dark room with bright windows. The complexity of the scene types contained in this dataset greatly increases the difficulty of recognition and detection, and it is difficult for unimproved general-purpose target detection methods to work well enough on this dataset. At the same time, the ExDark training dataset is too small, which affects the performance of the model, so there is a great need to expand the ExDark dataset to improve the generalization ability of the model.

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

3.2. Low-Level Image Enhancement

Most of the current publicly available low-light image datasets are from the image enhancement domain, where the images have no detection annotation information and the number of them is difficult to support the training of shimmering image detection models. In order to increase the available datasets, we will expand the datasets by collecting 5000 low-light images from the real world and generate more low-light images by combining low-order image enhancement techniques. Three low-cost image enhancement methods will be used to process the ExDark and reality images, namely contrast adjustment, enlarging target, and super resolution.

Contrast Adjustment: In this paper, the Histogram Equalization (HE) method is used to adjust the contrast of an image. This method makes the distribution of grey levels in the histogram more uniform, so that the dynamic range is expanded, thus enhancing the contrast of the whole image [23].

Assuming that the grey level distribution of the very first image is f, g represents the grey level distribution of the original image after the HE method. Let the grey levels in the grey level distribution be r and s, and their corresponding numbers of pixels be f_r and g_s,

respectively. Then the probability of the grey level distribution is given by Equation (1), and D stands for the total grey levels.

$$p_f(r) = \frac{f_r}{\int\limits_{r \in D} f(r)dr}, p_g(s) = \frac{g_s}{\int\limits_{s \in D} g_s} \tag{1}$$

Equation (2) is derived from Equation (1):

$$\int p_f(r)dr = \int p_g(s)ds = 1 \tag{2}$$

In order to make the histogram uniformly distributed, it is necessary to make the distribution probability of each grey level equal, and the derivation yields Equation (3):

$$p_g(s) = \frac{1}{g_{max} - g_{min}} \tag{3}$$

Bringing Equation (3) into Equation (2) leads to Equation (4):

$$\int_{f_{min}}^{f} p_f(r)dr = \int_{g_{min}}^{g} \frac{1}{g_{max} - g_{min}} ds \tag{4}$$

This gives the transformation Equation (5) for the HE method, where $p_f(r)$ is the distribution of the original map.

$$g = (g_{max} - g_{min})p_r(r) + g_{min} \tag{5}$$

Enlarging Target: To better extract the subject features in low-light images, we use a modified seam carving algorithm to magnify the subject. The seam carving algorithm's main feature is that it takes into account the importance of the content of the image, which defines the importance of the different scenes in the image through the energy function [24]; then, by constantly deleting or copying the pixel lines that have the lowest energy, at the same time, it maintains the original appearance of the visual subject at the maximum extent. The original appearance of the visual subject at the same time accomplishes a non-equal aspect ratio zoom.

Let i be an $n * m$ image with vertical pixel lines defined as follows:

$$S^x = \{S_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, s, t, \forall i, |x(i) - x(i-1) \leq 1| \tag{6}$$

Here, $x(i)$ is the mapping: $x : [1, \ldots, n] \rightarrow [1, \ldots, m]$. From the above equation, it can be seen that the vertical pixel line is an 8-connected path consisting of pixels from the first to the last row of the image, with one and only one pixel removed from each row. Similarly, the horizontal pixel line is defined as:

$$S^y = \{S_j^y\}_{j=1}^m = \{(x(j), j)\}_{j=1}^m, s, t, \forall j, |x(j) - x(j-1) \leq 1| \tag{7}$$

where $x(j)$ is the mapping: $y : [1, \ldots, m] \rightarrow [1, \ldots, n]$. Horizontal pixel lines are 8-connected paths consisting of pixels from the first to the last column of the image, with one and only one pixel removed from each column. Thus, the path of the vertical pixel line is $I_s = \{I(s_i)\}_{j=1}^n = \{I(x(i), i)\}_{i=1}^n$. Then the path of the horizontal pixel line has a similar situation. If a row or column of an image is to be deleted, then processing an image results in all pixels below or to the right of the line being shifted up or to the left to complement the deleted pixel line.

Line cropping has a limited effect on the visual appearance of an image; it only affects the area around the line being deleted or added, and has no effect at all on the remaining pixels in the image. Therefore, it is crucial to find the appropriate pixel line. Knowing

that the energy function of each pixel of an image is e, the total energy of the pixel line is defined as

$$E(s) = E(I_s) = \sum_{i=1}^{n} e(I(s_i)) \tag{8}$$

Find the vertical pixel line with the lowest total energy:

$$S^* = \min E(s) = \min_s \sum_{i=1}^{n} e(I(s_i)) \tag{9}$$

The horizontal pixel lines are found in a similar way. The finding of the lowest energy pixel line is solved using the dynamic programming method. For a pixel point (i, j) in image I, remembering that the cumulative energy of the point is $M(i, j)$, we have

$$\begin{cases} M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j+1)), & i \neq 1 \\ M(i,j) = e(i,j), & i = 1 \end{cases} \tag{10}$$

Therefore, it is only necessary to traverse the second to last row in image I and calculate the above equation to derive the cumulative energy. The pixel point in the last row with the smallest cumulative energy is the end point of the vertical pixel line with the smallest energy. Then, we go back from this point, each time looking for the pixel point with the smallest cumulative energy in the field previous to the known minimum cumulative energy point, and so on, until the vertical pixel line with the smallest energy is found.

For finding horizontal pixel lines, the method is similar and has cumulative energy $M(i, j)$, as in the following equation:

$$\begin{cases} M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j+1)), & j \neq 1 \\ M(i,j) = e(i,j), & j = 1 \end{cases} \tag{11}$$

As can be seen from the formula for cumulative energy, the minimum of the last line of M must be the end of the vertical minimum energy line, so the vertical minimum energy line can be found by backtracking from this point. The method of finding the horizontal minimum energy line is similar.

To make the pixel importance calculation more reasonable, the original algorithm is improved by the following process: multiplying the gradient map with the visual saliency map to obtain the final importance energy map of the whole image, calculating the cumulative energy map of the image, and then searching for the lowest energy point, going back to find the lowest energy line, and deleting or inserting the lowest energy line to obtain the final target image. The improved algorithm can better preserve the subject effect, especially for visual subjects in dark lighting conditions.

Super Resolution: In this paper, we use the SwinIR network model based on Transformer to reconstruct the image with ultra-high resolution, which consists of three modules: shallow feature extraction, deep feature extraction, and reconstruction module [25]. The shallow feature extraction module uses a convolutional layer to extract shallow features, which are passed directly to the reconstruction module to preserve low-frequency information. The deep feature extraction module consists mainly of residual Swin Transformer Blocks (RSTBs) containing Swin Transformers, and each RSTB uses multiple Swin Transformer layers for local attention and cross-window interaction. In addition, SwinIR adds a convolutional layer at the end of the block for feature enhancement and uses a residual link to provide a shortcut for feature aggregation. Finally, shallow and deep features are fused in the reconstruction module to achieve high-quality image reconstruction. The Swin Transformer layer is based on the standard multi-head self-attention of the original Transformer layer. The main difference is the self-attention and shift window mechanism. Given an input of size $W \times H \times C$, the Swin Transformer first transforms the input to $(HW/M2) \times M2 \times C$ and divides it into one window with dimensions $M \times M$ and without overlapping, where $HW/M2$ is the total number of windows. Then, the self-attention

of each window is calculated separately. For a local window feature $X \in R^{M^2 \times C}$, the query vector Q, the key vector K, and the value vector V are computed as follows (12):

$$Q = XP_Q, K = XP_K, V = XP_V \tag{12}$$

where P_Q, P_K, and P_V are the learnable matrices of the query vectors, the key vectors, the value vectors, and share weights between the windows, respectively, and in general, $Q, K, V \in R^{M^2 \times d}$ through the self-attention mechanism within a local window is calculated as can be seen below. The attention matrix is shown in Equation (13):

$$Attention(Q, K, V) = softMax(\frac{QK^T}{\sqrt{d}} + B)V \tag{13}$$

where B is the learnable relative positional encoding. $d = \frac{C}{N_h ead}$, and $N_h ead$ represents the amount of attention from multiple heads. Thereafter, we execute the attention function h times in parallel and stitch together the results of the multi-headed self-attention mechanism.

A multi-layer perception containing two fully connected layers is then used, with a Gelu nonlinear function between the fully connected layers, which increases the nonlinear expressiveness of the model. In addition, the multi-head attention using residual connections and the multi-layer LayerNorm layer is added before the two modules of multi-head attention and multi-layer perception using residual connection. The whole process is expressed in Equations (14) and (15):

$$X = MSA(LN(X)) + X \tag{14}$$

$$X = MLP(LN(X)) + X \tag{15}$$

In summary, we augmented the images using three lower-order image processing techniques with the aim of expanding the training dataset to improve the performance of the higher-order visual model, and Figure 2 shows some of the processed images.

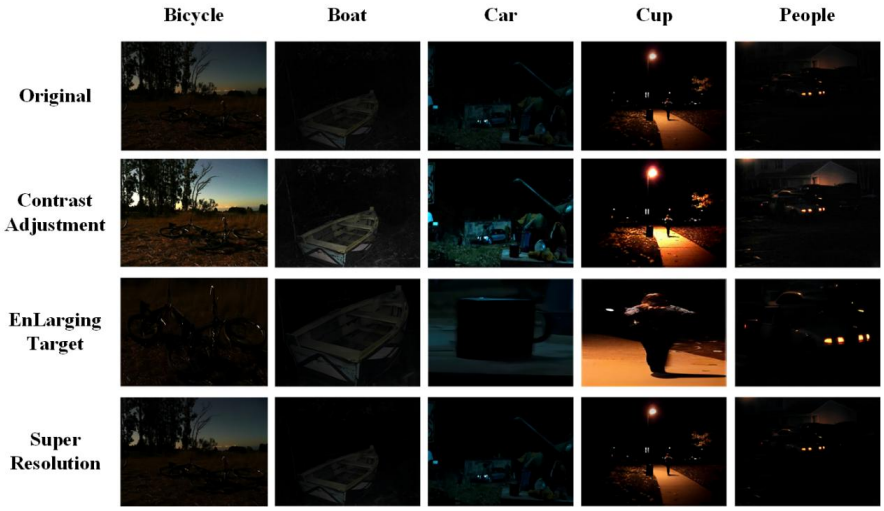

Figure 2. Image display after low-level data enhancement.

4. Implementation of Model Deployment on MPSoC

In this section, we use the Vitis-AI development environment [26] to implement the deployment of object detection models onto MPSoCs and the implementation process is shown in Figure 3. The Vitis-AI development environment consists primarily of the Vitis-AI Development Kit (DNNDK) for AI inference on Xilinx ZYNQ series hardware platforms at the edge or in the cloud, consisting of optimized DPU IP cores, tool libraries,

runtime libraries, and a library of deep neural network (DNN) models. DNNDK consists of optimized DPU IP cores, tool libraries, runtime libraries, and a library of DNN models. The tools provided by Vitis-AI call the DPU IP cores on the PL side, making it easy to develop DNN inference-based applications on the object board. Currently, the deployment of DNN on hardware platforms supports the TensorFolw framework based on the Python language and the Caffe framework based on the C++ language. In this paper, the object recognition model is developed based on DNN of TensorFlow, and the object board model is ZU3EG-AXU3EGB.

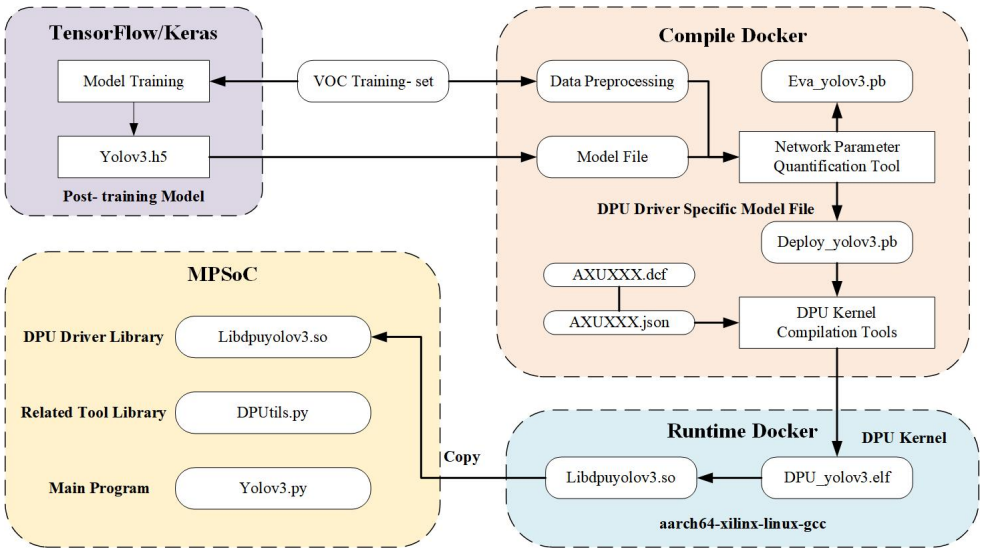

Figure 3. Implementation process of model deployment on MPSoC.

In the above process, DNN parameter quantization calibration and DPU kernel compilation are key to achieving model-to-MPSoC deployment. Quantization calibration is the process of converting the parameters of a model from high-precision quantities to low-precision quantities, thereby reducing the storage space and memory footprint of the model. In this paper, the quantization calibration of the object detection model is performed to speed up the computation and reduce the power consumption of the device, making it suitable for MPSoC deployment. Once the neural network has been trained in the Mainstream framework, the network model can be exported as a .pb file containing the model's parameter information. Before the DPU imports this parameter information, it must first be quantized, i.e., fixed-point. The fixed-point process requires the input of an unlabelled test set. Test data are required for post-quantization calibration to recover or optimize accuracy degradation due to information loss during the quantization process, and to improve model generalization. Once the quantized .pb model is obtained, the DPU kernel compilation phase will proceed. The DPU kernel is compiled using the Vitis AI compiler. During the compilation process, the Vitis AI compiler converts the pre-processed deep learning model IR into binary code and stores it in an .elf file. This file can be loaded and executed by the DPU for efficient computation and inference tasks.

After obtaining the .elf file, the next step is to cross-compile it, which is used to generate the dynamic link library .so file. It contains the YOLOv3 kernel and only needs a Python application driver to perform the YOLOv3 extrapolation on the object board. The 64-bit ARM GCC cross-compile toolchain in the Xilinx Runtime docker is used to generate the dynamic link library files. This is followed by setting up the object board to boot and initialize the Petalinux system image, and configuring the runtime environment. At this point, the full process of object detection model deployment to an MPSoC implementation is in place.

The ARM side of the object board uses the DNNDK API (Python) to input the pre-processed image into the DPU, and obtains the data after the convolution operation of the DPU through the API. Figure 4 shows the flow of the object board for edge computing tasks.

Place the compiled dynamic link library .so file in the same directory as the application, use the DNNDK API to parse the .so file, make the DPU refer to the library file to run in a predetermined way, and then use the API to retrieve the data from the memory allocated by the DPU. Finally, display the coordinates and category information of the object obtained after processing.

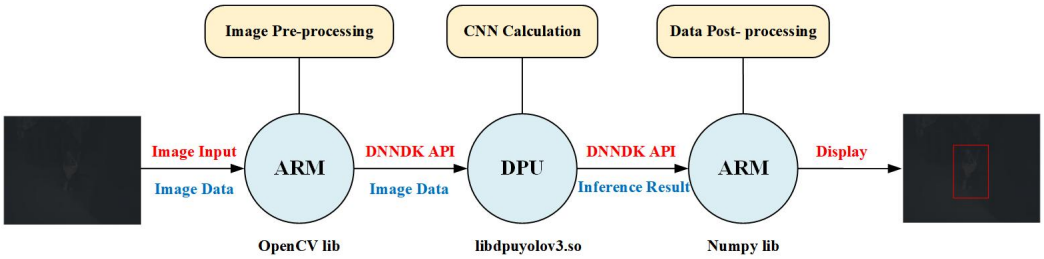

Figure 4. Object detection process based on MPSoC.

5. Experiment Results and Analysis

5.1. Experiment Setup

The hardware configuration used in the experiment is AMD 2700X, Nvidia GeForce GTX 2070 graphics card, Linux operating system; the software environment is CUDA11.1.0, Cudnn8.0, and the deep learning framework is TensorFlow. The training process adopts the SGD optimizer, the initial learning rate is 0.02, the momentum size is 0.9, the batchsize is 8, and the number of iterations is 150. The initial learning rate is 0.02, the momentum size is 0.9, the batch size is 8, and the number of iterations is 150.

5.2. Datasets and Evaluation Indicators

5.2.1. Datasets

FLIR dataset: The FLIR dataset [27] was released by FLIR in 2018 and contains 10 k manually annotated thermal infrared images and their corresponding reference RGB images acquired during day and night. The FLIR dataset is an image-unaligned multispectral multi-object detection dataset consisting of RGB images captured by the FLIR BlackFly RGB camera with a resolution of 1280 × 1024 and thermal infrared images captured by the FLIR Tau2 thermal camera with a resolution of 640 × 512. LLVIP dataset: Jia et al. [28] proposed the LLVIP dataset, which contains a large number of street-level images of pedestrians and cyclists captured by a multispectral camera with an overhead surveillance view. Most of the images were taken in very dark scenes. In addition, we also used the Exdark dataset and real-life low-light images for our experiments, which are described in Section 3.

5.2.2. Evaluation Indicators

In this paper, the FOLD algorithm is compared with other methods to measure its performance. The experimental models in this paper are all evaluated using the target detection metric proposed by the MS-COCO dataset: mean average precision (mAP) [19,29,30]. LOP mAP50, mAP75, and mAP denote the average of all AP values for all categories when the intersection over union (IoU) is 0.50, 0.75, and 0.50:0.95, respectively. In addition, this paper compares them in terms of detection performance and computational performance. Detection performance is measured in frames per second (FPS). Computational performance is compared in terms of computational efficiency, processing time, and power consumption. Computational efficiency is defined as the number of image processes per unit time.

5.3. Experimental Results and Analysis

5.3.1. Comparison of Detection Performance

In this section, we compare the detection performance of the FOLD proposed in this paper with other methods on FLIR, LLVIP, ExDark, and real image datasets. The algorithms

compared include eight algorithms such as YOLOv3 based on unimodal RGB images, YOLOv5 based on unimodal RGB images, CFT [31], CCIFNet [32], LIME + YOLOv3, LIME + YOLOv5, DCE_ZERO + YOLOv3, and DCE_ZERO + YOLOv5. The first two are simple object detection models, while the last six are object detection models with image enhancement capabilities, including LIME and DCE_ZERO [33], which are two low-light image enhancement methods. From Tables 1–4, it can be seen that the FOLD algorithm achieves the highest detection results under the mAP50, mAP75, and mAP evaluation metrics. Although it is about the same as the dedicated LIME and DCE_ZERO methods, it still has a slight advantage. In general, the FOLD algorithm proposed in this paper achieves satisfactory detection results under all IoU thresholds, indicating that the method can be well generalized to different IoU thresholds. This shows that the method can be well generalized to different types of images.

Table 1. Performance comparison of different algorithms on FLIR dataset.

Method	Backbone	Data	mAP50	mAP75	mAP
YOLOv3	DarkNet	RGB	0.834	0.373	0.421
YOLOv5	CSPDarkNet	RGB	0.893	0.507	0.503
CFT	CSPDarkNet	RGB+T	0.955	0.717	0.628
CCIFNet	ResNet50	RGB+T	0.969	0.736	0.651
LIME + YOLOv3	DarkNet	RGB	0.959	0.741	0.659
LIME + YOLOv5	CSPDarkNet	RGB	0.961	0.743	0.660
DCE_ZERO + YOLOv3	DarkNet	RGB	0.955	0.738	0.656
DCE_ZERO + YOLOv5	CSPDarkNet	RGB	0.951	0.732	0.654
FOLD	DarkNet	RGB	0.971	0.743	0.662

Table 2. Performance comparison of different algorithms on LLVIP dataset.

Method	Backbone	Data	mAP50	mAP75	mAP
YOLOv3	DarkNet	RGB	0.850	0.369	0.431
YOLOv5	CSPDarkNet	RGB	0.897	0.524	0.502
CFT	CSPDarkNet	RGB + T	0.971	0.722	0.638
CCIFNet	ResNet50	RGB + T	0.970	0.728	0.631
LIME + YOLOv3	DarkNet	RGB	0.963	0.761	0.621
LIME + YOLOv5	CSPDarkNet	RGB	0.970	0.758	0.603
DCE_ZERO + YOLOv3	DarkNet	RGB	0.961	0.752	0.591
DCE_ZERO + YOLOv5	CSPDarkNet	RGB	0.943	0.697	0.593
FOLD	DarkNet	RGB	0.972	0.773	0.651

Table 3. Performance comparison of different algorithms on ExDark dataset.

Method	Backbone	Data	mAP50	mAP75	mAP
YOLOv3	DarkNet	RGB	0.839	0.368	0.429
YOLOv5	CSPDarkNet	RGB	0.918	0.539	0.525
CFT	CSPDarkNet	RGB + T	0.972	0.727	0.638
CCIFNet	ResNet50	RGB + T	0.974	0.727	0.645
LIME + YOLOv3	DarkNet	RGB	0.966	0.756	0.684
LIME + YOLOv5	CSPDarkNet	RGB	0.976	0.760	0.688
DCE_ZERO + YOLOv3	DarkNet	RGB	0.969	0.750	0.677
DCE_ZERO + YOLOv5	CSPDarkNet	RGB	0.971	0.752	0.674
FOLD	DarkNet	RGB	0.977	0.753	0.681

Table 4. Performance comparison of different algorithms on Real-image dataset.

Method	Backbone	Data	mAP50	mAP75	mAP
YOLOv3	DarkNet	RGB	0.869	0.380	0.434
YOLOv5	CSPDarkNet	RGB	0.910	0.521	0.511
CFT	CSPDarkNet	RGB + T	0.968	0.723	0.632
CCIFNet	ResNet50	RGB + T	0.965	0.719	0.632
LIME + YOLOv3	DarkNet	RGB	0.969	0.764	0.671
LIME + YOLOv5	CSPDarkNet	RGB	0.971	0.776	0.674
DCE_ZERO + YOLOv3	DarkNet	RGB	0.962	0.767	0.669
DCE_ZERO + YOLOv5	CSPDarkNet	RGB	0.968	0.769	0.672
FOLD	DarkNet	RGB	0.976	0.776	0.678

5.3.2. Comparison of Computational Performance

In this section, we deploy FOLD on boards such as FPGA MPSoC, RK3588, Raspberry Pi, and Atlas to perform target detection on images from FLIR, LLVIP, ExDark, and Real-image datasets, respectively, to compare the computational performance. A material object of an FPGA MPSoC for target detection based on FOLD is shown in Figure 5.

Figure 5. Material object of target detection system.

As can be seen from Table 5, the average detection rates of the FOLD deployed on FPGA MPSoC, RK3588, Raspberry Pi, and Atlas are 87.25, 45.75, 10.5, and 16.25, respectively. The detection rate deployed on FPGA MPSoC completely exceeds the 25 FPS rate and is much higher than the other few detection rates, which satisfies the naked eye observation without delay. In general, the deployment of the FOLD algorithm proposed in this paper on FPGA MPSoC achieves satisfactory detection rates, which indicates that the FPGA-based implementation of the target detection model can be applied to most of the downstream edge vision tasks.

Table 5. Comparison of detection rate in frames per second (FPS).

Datasets / Onboard	FLIR	LLVIP	ExDark	Real-Image	Avg.
FPGA MPSoC	92	89	85	83	87.25
RK3588	49	46	45	43	45.75
Raspberry Pi	12	11	11	8	10.5
Atlas	18	16	16	15	16.25

The average computational efficiency (CE), processing time (PT), and power consumption (PC) of FOLD deployed on FPGA MPSoC, RK3588, Raspberry Pi, and Atlas are shown

in Table 6. It can be seen that the power consumption of FOLD deployed on FPGA MPSoC is not the lowest, but the processing time is much lower than the time deployed on other devices. Therefore, the deployment of the FOLD algorithm proposed in this paper on FPGA MPSoC achieves satisfactory computational efficiency, which indicates that the deployment scheme has certain advantages to achieve high processing efficiency.

Table 6. Comparison of computation performance on different platforms.

Onboard	CE	PT (ms)	PC (W)
FPGA MPSoC	12.46	11.45	7.0
RK3588	6.35	21.86	7.2
Raspberry Pi	1.64	95.23	6.4
Atlas	2.95	61.54	5.5

5.3.3. Limitations

To investigate the robustness and limitations of the proposed method, we performed real-world tests in an external field with different illumination levels. When the illumination is lower than 0.01 lux, the proposed microbiology-enhanced target detection method will be more limited, and it is easy to have the problem that the target cannot be detected or the dark part will be circled out; at this time, the detection accuracy objectively increases and the system robustness benefits images of poor quality.

6. Conclusions and Future Work

In this paper, we explore and investigate object detection and related problems in low-light scenes, and propose a low-level image enhancement method for low-light object detection based on MPSoC. Aiming at the current problem of the lack of low-light image datasets and its poor quality, three low-order image enhancement methods are adopted to expand the dataset and improve the detection performance of YOLO in low-light scenes. Aiming at the problem of the limited improvement of the object detection performance of the existing low-light image enhancement methods, an edge object detection system based on the deployment of MPSoC is realized, which effectively improves the detection efficiency of the edge end. In future work, we will use more low-light image enhancement techniques for downstream visual tasks and explore a wider range of image evaluation metrics to comprehensively evaluate the effectiveness of different image enhancement techniques and their positive gain on object detection performance.

Author Contributions: Conceptualization, X.L. and Z.L.; methodology, X.L. and L.Z.; software, Z.H. and X.L.; validation, L.Z. and Z.H.; writing—original draft preparation, X.L. and Z.L.; writing—review and editing, Z.H. and L.Z.; supervision, Z.L.; project administration, X.L.; funding acquisition, Z.L and Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Management Fund of North University of China: 2310700037HX, in part by the Guangzhou Municipal Science and Technology Project: SL2022A04J00404, in part by the Research Start-up Fund in Shanxi Province under Grant 980020066.

Data Availability Statement: Data is unavailable due to privacy or ethical restrictions.

Acknowledgments: The authors would like to thank the Editors and Reviewers for their contributions to our manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dai, J.; He, K.; Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 3150–3158.
2. Kang, K.; Li, H.; Yan, J.; Zeng, X.; Yang, B.; Xiao, T.; Zhang, C.; Wang, Z.; Wang, R.; Wang, X.; et al. T-CNN: Tubelets with convolutional neural networks for object detection from videos. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 2896–2907. [CrossRef]

3. Fu, X.; Zeng, D.; Huang, Y.; Liao, Y.; Ding, X.; Paisley, J. A fusion-based enhancing method for weakly illuminated images. *Signal Process.* **2016**, *129*, 82–96. [CrossRef]
4. Guo, X.; Li, Y.; Ling, H. LIME: Low-light image enhancement via illumination map estimation. *IEEE Trans. Image Process.* **2017**, *26*, 982–993. [CrossRef] [PubMed]
5. Tao, L.; Zhu, C.; Xiang, G.; Li, Y.; Jia, H.; Xie, X. LLCNN: A convolutional neural network for low-light image enhancement. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4.
6. Wang, D.; Niu, X.; Dou, Y. A piecewise-based contrast enhancement framework for low lighting video. In Proceedings of the 2014 IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), Wuhan, China, 18–19 October 2014; pp. 235–240.
7. Land, E.H. The retinex theory of color vision. *Sci. Am.* **1977**, *237*, 108–129. [CrossRef] [PubMed]
8. Godard, C.; Matzen, K.; Uyttendaele, M. Deep burst denoising. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 538–554.
9. Lore, K.G.; Akintayo, A.; Sarkar, S. LLNet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognit.* **2017**, *61*, 650–662. [CrossRef]
10. Wei, C.; Wang, W.; Yang, W. Deep Retinex decomposition for low-light enhancement. In Proceedings of the British Machine Vision Conference (BMVC), Newcastle, UK, 3–6 September 2018; pp. 1–12.
11. Zhu, M.; Pan, P.; Chen, W.; Yang, Y. EEMEFN: Low-light image enhancement via edge-enhanced multi-exposure fusion network. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 13106–13113.
12. da Costa, G.B.P.; Contato, W.A.; Nazare, T.S.; Neto, J.E.S.B.; Ponti, M. An empirical study on the effects of different types of noise in image classification tasks. *arXiv* **2016**, arXiv:1609.02781.
13. Yuan, C.; Hu, W.; Tian, G.; Yang, S.; Wang, H. Multi-task sparse learning with Beta process prior for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 423–429.
14. Kvyetnyy, R.; Maslii, R.; Harmash, V.; Bogach, I.; Kotyra, A.; Gradz, Z.; Zhanpeisova, A.; Askarova, N. Object Detection in Images with Low Light Condition. In *Photonics Applications in Astronomy Communications Industry and High Energy Physics Experiments, International Society for Optics and Photonics*; SPIE: Bellingham, WA, USA, 2017; Volume 10445, p. 104450W.
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
16. Ovtcharov, K.; Ruwase, O.; Kim, J.-Y.; Fowers, J.; Strauss, K.; Chung, E.S. Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Res. Whitepap.* **2015**, *2*, 1–4.
17. Nurvitadhi, E.; Venkatesh, G.; Sim, J.; Marr, D.; Huang, R.; Ong Gee Hock, J.; Liew, Y.; Srivatsan, K.; Moss, D.; Subhaschandra, S.; et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 22–24 February 2017; pp. 5–14.
18. Zhao, R.; Song, W.; Zhang, W.; Xing, T.; Lin, J.-H.; Srivastava, M.; Gupta, R.; Zhang, Z. Accelerating binarized convolutional neural networks with software-programmable FPGAs. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 15–24.
19. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
20. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
21. Everingham, M.; Gool, L.V.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **2009**, *88*, 303–338. [CrossRef]
22. Loh, Y.P.; Chan, C.S. Getting to know low-light images with the exclusively dark dataset. *Comput. Vis. Image Understand.* **2019**, *178*, 30–42. [CrossRef]
23. Wang, W.; Zhang, P.; Li, M.; Ren, S. FPGA Implementation of Video Enhancement Algorithm in Low Illumination Conditions. *Bandaoti Guangdian/Semiconduct. Optoelectron.* **2017**, *38*, 754–757.
24. Rubinstein, M.; Shamir, A.; Avidan, S. Improved seam carving for video retargeting. *Proc. ACM SIGGRAPH Pap.* **2008**, *27*, 1–9. [CrossRef]
25. Liu, Z.; Lin, Y.T.; Cao, Y.; Hu, H.; Guo, B.N. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10012–10022.
26. Xilinx: Vitis AI User Guide (UG1414). Available online: https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview/ (accessed on 13 March 2023).
27. FLIR: Flir Thermal Dataset for Algorithm Training. Available online: https://www.flir.in/oem/adas/adas-dataset-form/ (accessed on 29 December 2023).
28. Jia, X.; Zhu, C.; Li, M.; Tang, W.; Zhou, W. LLVIP: A visible-infrared paired dataset for low-light vision. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 3496–3504.

29. Chen, C.; Mo, J.; Hou, J.; Wu, H.; Liao, L.; Sun, W.; Yan, Q.; Weisi Lin, W. TOPIQ: A Top-down Approach from Semantics to Distortions for Image Quality Assessment. *arXiv* **2023**, arXiv:2308.03060.
30. Liao, L.; Xiao, J.; Wang, Z.; Lin, C.W.; Satoh, S. Guidance and Evaluation: Semantic-Aware Image Inpainting for Mixed Scenes. In *Computer Vision–ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; ECCV 2020, Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12372. [CrossRef]
31. Fang, Q.; Han, D.; Wang, Z. Cross-modality fusion transformer for multispectral object detection. *arXiv* **2021**, arXiv:2111.00273.
32. Yan, C.; Zhang, H.; Li, X.; Yang, Y.; Yuan, D. Cross-modality complementary information fusion for multispectral pedestrian detection. *Neural Comput. Appl.* **2023**, *35*, 10361–10386. [CrossRef]
33. Shu, Z.; Zhang, Z.; Song, Y. Low light image object detection based on improved YOLOv5. *Prog. Laser Optoelectron.* **2023**, *60*, 8. [CrossRef]