sensors

MDPI

Article

A Lightweight Pedestrian Detection Engine with Two-Stage Low-Complexity Detection Network and Adaptive Region Focusing Technique

Luying Que, Teng Zhang, Hongtao Guo, Conghan Jia, Yuchuan Gong, Liang Chang and Jun Zhou *

School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 201821010433@std.uestc.edu.cn (L.Q.); 202021010224@std.uestc.edu.cn (T.Z.); ght_work@163.com (H.G.); 201822010437@std.uestc.edu.cn (C.J.); 201911012135@std.uestc.edu.cn (Y.G.); liangchang@uestc.edu.cn (L.C.)

* Correspondence: zhouj@uestc.edu.cn

Abstract: Pedestrian detection has been widely used in applications such as video surveillance and intelligent robots. Recently, deep learning-based pedestrian detection engines have attracted lots of attention. However, the computational complexity of these engines is high, which makes them unsuitable for hardware- and power-constrained mobile applications, such as drones for surveillance. In this paper, we propose a lightweight pedestrian detection engine with a two-stage low-complexity detection network and adaptive region focusing technique, to reduce the computational complexity in pedestrian detection, while maintaining sufficient detection accuracy. The proposed pedestrian detection engine has significantly reduced the number of parameters (0.73 M) and operations (1.04 B), while achieving a comparable precision (85.18%) and miss rate (25.16%) to many existing designs. Moreover, the proposed engine, together with YOLOv3 and YOLOv3-Tiny, has been implemented on a Xilinx FPGA Zynq7020 for comparison. It is able to achieve 16.3 Fps while consuming 0.59 W, which outperforms the results of YOLOv3 (5.3 Fps, 2.43 W) and YOLOv3-Tiny (12.8 Fps, 0.95 W).

Keywords: pedestrian detection; lightweight; neural network; adaptive; FPGA

Citation: Que, L.; Zhang, T.; Guo, H.; Jia, C.; Gong, Y.; Chang, L.; Zhou, J. A Lightweight Pedestrian Detection Engine with Two-Stage Low-Complexity Detection Network and Adaptive Region Focusing Technique. *Sensors* **2021**, *21*, 5851. https://doi.org/10.3390/s21175851

Academic Editors: Paweł Pławiak and Yitzhak Yitzhaky

Received: 8 June 2021
Accepted: 23 August 2021
Published: 30 August 2021

1. Introduction

Pedestrian detection has been widely used in applications such as autonomous driving, video surveillance, and intelligent robots. In the past, the pedestrian detection was realized by traditional image processing and machine learning methods, such as the histogram of oriented gradient (HOG) and the support vector machine (SVM). The HOG is a classic feature descriptor used in image processing, which can effectively extract the features of pedestrians and send them to classifiers, such as SVM, to realize pedestrian detection [1]. Later, a discriminatively trained part-based models (DPM) detection method is proposed, which improves the detection accuracy by constructing the excitation template of each component, such as an edge and corner feature, then determines the target position according to the distribution of the excitation [2]. Wavelet domain-based pedestrian detection methods have also been proposed, for instance, a multi-feature (Multiftr) detection method is proposed, which extracts Haar-like wavelets features for the detection [3]. To further improve the detection accuracy, an aggregate channel features (ACF) detection method is proposed, which classifies aggregated channel features by decision tree [4]. As the features of pedestrians become more and more complex, such as appearance, clothing, dress, posture, lighting, background, and image resolution, it becomes increasingly challenging for the traditional methods to extract the effective features and achieve high detection accuracy. Also, it takes significant engineering effort to find suitable features for the detection.

To address the above issues, deep learning-based pedestrian detection methods have been proposed. Compared to the traditional methods, using signal processing and machine

learning, these methods are able to extract the complex features of pedestrians more effectively and automatically through an end-to-end neural network, which leads to a higher detection accuracy and reduced effort for feature engineering. There are some scholars using a combination of machine learning- and deep learning-based methods. For example, in [5], a pedestrian method, which used SVM and CNN, is proposed. In this method, a motion detection method is used to locate the interested area, then principal component analysis and support vector machines are used to extract the textural feature vector and filter out interference regions, and, at last, a CNN is used to execute the pedestrian classification. Besides, many deep learning-based pedestrian methods have been proposed, such as active detection module (ADM) [6], multi-scale deep convolutional neural network (MS-CNN) [7], and graininess-aware deep feature learning (GDFL) [8]. These methods significantly improve the detection accuracy. In addition to the dedicated pedestrian detection methods, some methods are proposed by modifying the general object detection methods. For instance, pedestrian detection methods [9,10] based on YOLO and pedestrian methods [11,12] based on Faster R-CNN. However, while achieving a higher detection accuracy, the deep learning-based methods significantly increase the computational complexity, and cause high power consumption and a large processing time, making them unsuitable for hardware- and power-constrained mobile applications, such as drones and mobile robots.

In this work, we propose a lightweight pedestrian detection engine with a two-stage low-complexity neural network and adaptive region focusing technique, for power- and resource-constrained intelligent video surveillance applications, such as drone-based surveillance.

The main contribution of this work can be summarized as follows:

- A two-stage low-complexity neural network for pedestrian detection is proposed, which significantly reduces the number of parameters and operations of the detection neural network, while maintaining a high detection accuracy;
- An adaptive region focusing technique is proposed, which further reduces the computational complexity by removing the redundancy in the pedestrian detection in video streams;
- The proposed lightweight pedestrian detection engine has been implemented on a Xilinx FPGA Zynq7020, to evaluate its performance and power consumption.

2. Related Work

In the early stages, pedestrian detection was performed by extracting various features in the input image and sending them to a classifier for detection. For example, in [1], it is proposed to extract the HOG features of the image and use an SVM classifier for the pedestrian detection. To improve the accuracy, a latent SVM classifier is proposed and used with the HOG features for the pedestrian detection [13], which achieves a higher detection accuracy than using the conventional SVM classifier. In [14], it is proposed to combine the HOG features with the local binary pattern (LBP) features for the pedestrian detection. Other than the HOG-related methods, in [15], it is proposed to extract the gradient responses of the input image in different directions, and compute the local average of these responses around each pixel, to build shapelets features. The features are then sent to an Adaboost classifier for pedestrian detection. However, this method only relies on the edge features, which is not very efficient. In [16], the edge features are combined with the texture and color features to improve the detection accuracy.

In the recent years, deep learning-based methods have been heavily used in pedestrian detection, due to their high detection accuracy. Many works used the semantic information of a pedestrian for the detection. For example, the part and context network (PCN), using the semantic and contextual information of the body parts of the pedestrian [17]. Liu proposed a pedestrian detection method using the semantic labeling with the traditional HOG + SVM method, to improve the detection accuracy [18]. In pedestrian detection, the detection of small-scale pedestrians and occluded pedestrians are two major challenges. A

pedestrian detection network (PedJointNet) is proposed that uses the head–shoulder feature of pedestrians as a complement to full-body prediction, to boost the detection accuracy [19]. In [20], a location bootstrap module and a semantic transition module are proposed to improve the detection accuracy of small-scale and occluded pedestrians. The location bootstrap is used to address predicted bounding boxes with relatively worse location precision, and the semantic transition module is used to extract more contextual information to relieve the semantic inconsistency of the skip-layer fusion for the detection of occluded pedestrians. Pedestrian detection in low-resolution images is also a challenge. In [21], a multi-resolution generative adversarial network (MRGAN) is proposed to simultaneously conduct multiresolution pedestrian detection, by generating high-resolution pedestrian images from low-resolution images. In [22], a fused discriminative metric learning (F-DML) approach is proposed to learn the optimal Mahanalobis metric, which transforms the low-resolution feature space into a new classification space, to improve the detection accuracy for low-resolution images. In addition, some works focus on pedestrian detection in low-light conditions. For example, a guided deep network is proposed to learn the extraction of multi-modal-like features from a single data modality in the framework of knowledge distillation via teacher–student training [23]. The teacher network, trained using the multi-modal data of RGB and thermal images, guides the student network to extract RGB and thermal-like features from RGB images alone. The multiScale detection network (MSDN) has also been proposed to solve the low-light problem [24]. In this method, a deep neural network is used to learn a non-linear mapping between RGB and thermal data. The learned feature representations are then transferred to another deep network, which receives the RGB image and generates the detection result.

Compared with the conventional machine learning methods, the deep learning-based methods achieve a higher detection accuracy. However, the computational complexity, including the number of parameters and operations, has been significantly increased. To address this issue, some low-complexity pedestrian detection methods have been proposed [25,26]. These works propose a lightweight neural network with a reduced number of parameters and operations. However, the reduction is still limited and/or the detection accuracy is heavily affected.

3. Proposed Pedestrian Detection Engine

In this work, a low-complexity pedestrian detection method has been proposed for power- and resource-constrained intelligent video surveillance applications. In this method, a two-stage low-complexity pedestrian detection network (TLDN) is proposed, to reduce the number of parameters and operations in deep learning-based pedestrian detection. An adaptive region focusing technique (ARFT) is proposed to further reduce the number of operations, by utilizing the feature of pedestrian detection in a video stream. Figure 1 shows the overall architecture of the proposed pedestrian detection method.

3.1. Proposed Two-Stage Low-Complexity Pedestrian Detection Network

The details of the proposed two-stage low-complexity pedestrian detection network are shown in Figure 2. The first stage of the detection network is used to generate the initial bounding boxes (bbox) of the pedestrians. Before the network, pyramid images are generated to obtain different scales of the input image. The first-stage network is a fully convolution network and, therefore, the size of the input image is flexible. As shown in Figure 2, the first-stage network contains only six convolution layers. As the network is shallow, in order to improve the efficiency of the feature extraction of pedestrians, rectangular-shaped kernels, instead of square-shaped kernels, are used for the convolution. We have conducted some experiments regarding the shape of the convolution kernel, as shown in Figure 3. It can be observed that with square-shaped convolution kernels, regardless of the kernel size, the precision is below 68% and the miss rate is above 64%. After changing the kernel shape to rectangular, the precision is improved to 85.18% and the miss rate is reduced to 25.15%. All the convolution layers in the first-stage network use

rectangular-shaped kernels with different sizes, as shown in Table 1 and Figure 2, where CK, CS, PK, and PS are the convolution kernel size, convolution stride, max pooling kernel size, and max pooling stride, respectively. The outputs of the first network include the offset values of the obtained bboxes and their confidence scores.

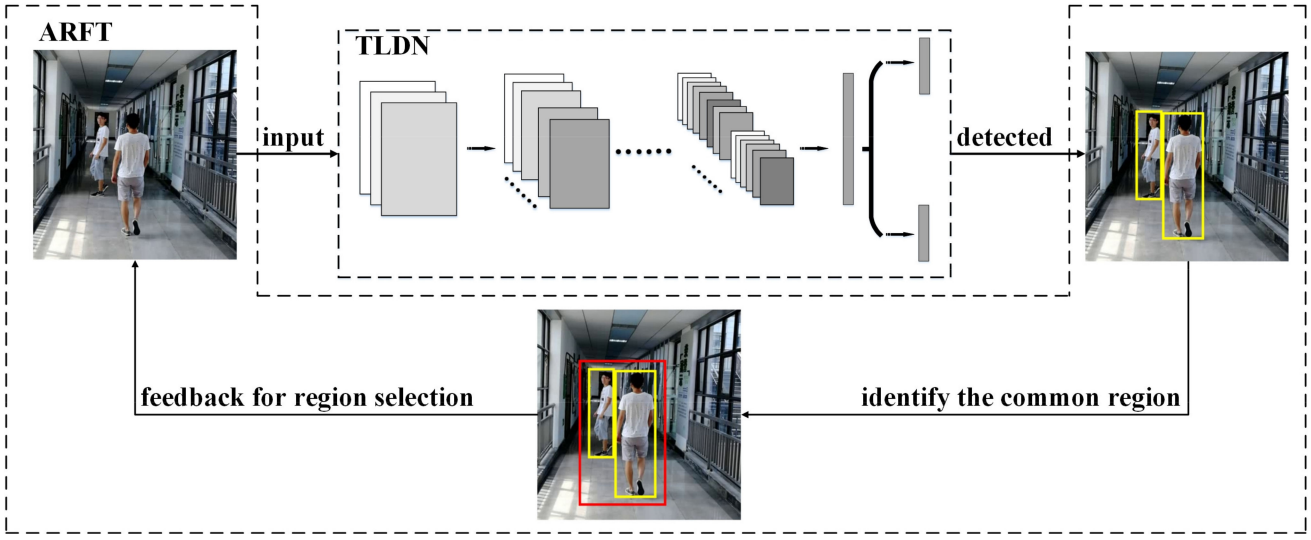

Figure 1. Overall architecture of the proposed method.

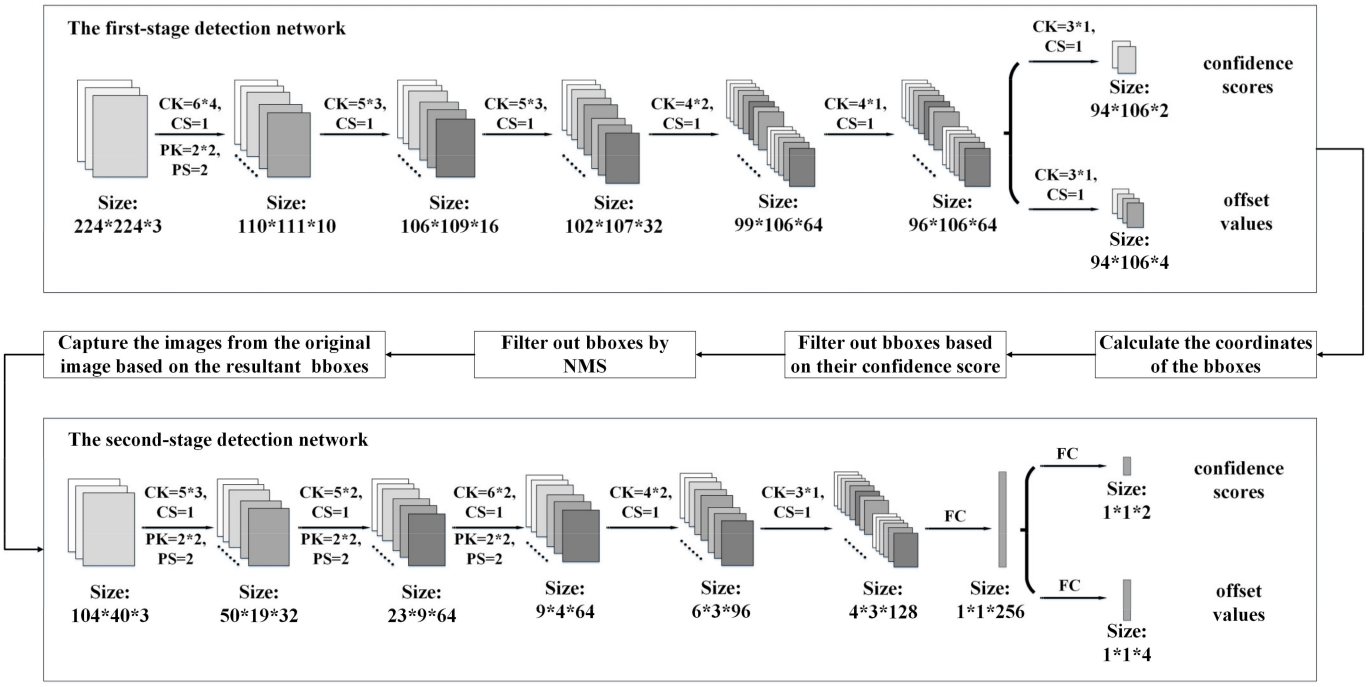

Figure 2. Proposed TLDN. The upper part is the first-stage detection network and the lower part is the second-stage detection network.

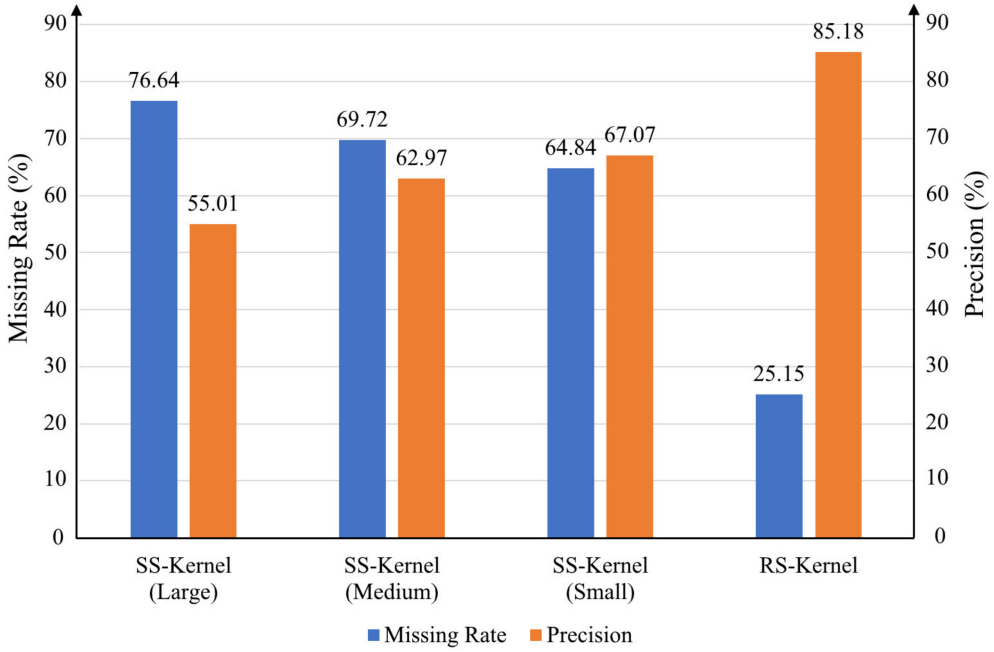

Figure 3. Comparison of experiment results of different kernels. SS-Kernel means square-shaped kernel and RS-Kernel means rectangular-shaped kernel.

Table 1. The values and specific names of CK, CS, PK and PS in every layer.

Stage	Layer	Convolution Kernel Size (CK)	Convolution Stride (CS)	Max Pooling Kernel Size (PK)	Max Pooling Stride (PS)
First stage	1	6 * 4	1	2 * 2	2
	2	5 * 3	1	/	/
	3	5 * 3	1	/	/
	4	4 * 2	1	/	/
	5	4 * 1	1	/	/
	6	3 * 1	1	/	/
Second stage	1	5 * 3	1	2 * 2	2
	2	5 * 2	1	2 * 2	2
	3	6 * 2	1	2 * 2	2
	4	4 * 2	1	/	/
	5	3 * 1	1	/	/

With the offset values of the bboxes, the coordinates of the bboxes can be calculated. Firstly, the coordinates of the base bboxes can be calculated using Equations (1)–(4).

$$x_1 = (s_w * id_x)/r \tag{1}$$

$$y_1 = (s_h * id_y)/r \tag{2}$$

$$x_2 = (s_w * id_x + c_w)/r \tag{3}$$

$$y_2 = (s_h * id_y + c_h)/r \tag{4}$$

where x_1, y_1, x_2, y_2 are the horizontal and vertical coordinates of the upper left corner and the lower right corner of the base bboxes, respectively. s_w is the ratio of the width of the input image to the width the output feature map, s_h is the ratio of the height of the input image height to the height of the output feature map, r is the ratio of the size of the input image to the size of the largest scale pyramid image, c_w is the allowed minimum width of the pyramid image, which is set to 15 (below this width, the pyramid image generation is stopped), and c_h is the allowed minimum height of the pyramid image, which is set to 39.

id_x is the horizontal index of each pixel of the output feature map, and id_y is the vertical index of each pixel of the output feature map.

After the coordinates of the base bboxes are calculated, the coordinates of the actual bboxes can be generated using Equations (5)–(8).

$$x_1' = x_1 + \Delta x_1 * (x_2 - x_1) \tag{5}$$

$$y_1' = y_1 + \Delta y_1 * (y_2 - y_1) \tag{6}$$

$$x_2' = x_2 + \Delta x_2 * (x_2 - x_1) \tag{7}$$

$$y_2' = y_2 + \Delta y_2 * (y_2 - y_1) \tag{8}$$

where x_1',y_1',x_2', y_2' are the horizontal and vertical coordinates of the upper left corner and the lower right corner of the actual bboxes after offset, respectively. Δx1, Δy1, Δx2, Δy2 are the offsets of the horizontal and vertical coordinates of the upper left corner and the lower right corner, respectively.

After that, the bbox selection is performed. First, some of the bboxes are filtered out based on their confidence score. A threshold is set to filter out the bboxes with a confidence score less than the threshold. Then, non-maximum suppression (NMS) is applied to remove the redundant bboxes, by selecting the bbox with the highest confidence score and calculating the intersection over union (IoU) value between it and all the candidate bboxes. The bboxes with IoU values higher than the threshold are removed. This process is iterated until no bbox can be removed.

The images corresponding to the resultant bboxes are sent to the second-stage network for computation one-by-one. The second stage of the detection network contains five convolution layers and two fully connected layers. All the convolution layers in the second-stage network use rectangular-shaped kernels with different sizes, as shown in Figure 2. The outputs of the second-stage network include the offset value of the obtained bbox and its confidence score for each image.

The second stage of the detection network is used to re-evaluate the initial bboxes. Its inputs are the images captured from the original image, based on the bboxes from the first stage. We will uniformly scale these images to 104×40, and then input them into the second-stage detection network. When the last layer of the second-stage detection network gets the output, we use Equations (5)–(8) to correct the bboxes represented by the input images.

The re-evaluated bboxes will be closer to the real situation. We filter all the re-evaluated bboxes, set a stricter score threshold, and then use the NMS algorithm to remove duplicate bboxes again. Through this stage of evaluation and filtering, we get the final pedestrian detection results.

3.2. Proposed Adaptive Region Focusing Technique

The proposed TLDN has small number of parameters; however, the second stage of the network needs to be computed multiple times, depending on the number of output bboxes from the first stage, which involves a large number of operations. To reduce the number of operations, we have proposed the ARFT technique, by utilizing the correlation among a sequence of frames in the video stream. The basic idea of the ARFT is to focus on the regions of the pedestrians in the image, and only perform the pedestrian detection in these regions for subsequent frames, to reduce the number of operations. The details of the technique are shown in Figure 4. When the pedestrians are detected in the input image, the regions of the pedestrians will be identified and used to guide the pedestrian detection for subsequent frames. A way to do this is to identify the regions for each detected pedestrian in the image and send them one-by-one to the detection network. However, this causes repeated computation of the detection network, especially when the number of regions is large, resulting in a limited reduction in the number of operations. To address this issue, we propose to identify the common region for all the detected pedestrians, as shown in

Figure 4, and send it to the detection network for the subsequent frame. We first find a region that just includes all the bboxes of the detected pedestrians. Then, this region is enlarged by increasing its height and width outwards by 20%. This acts as a safeguard area to avoid a miss detection for next time, as the pedestrians may be moving. In this way, the detection network only needs to be computed once. Although the common region method slightly increases the region size compared to the separate region method, the final number of operations is significantly reduced (as shown in the experimental results in Section 5–D). In addition, to avoid the miss detection of newly appearing pedestrians outside the common region, a full-region pedestrian detection is performed intermittently (e.g., every five frames), or when no pedestrian is detected in the current frame.

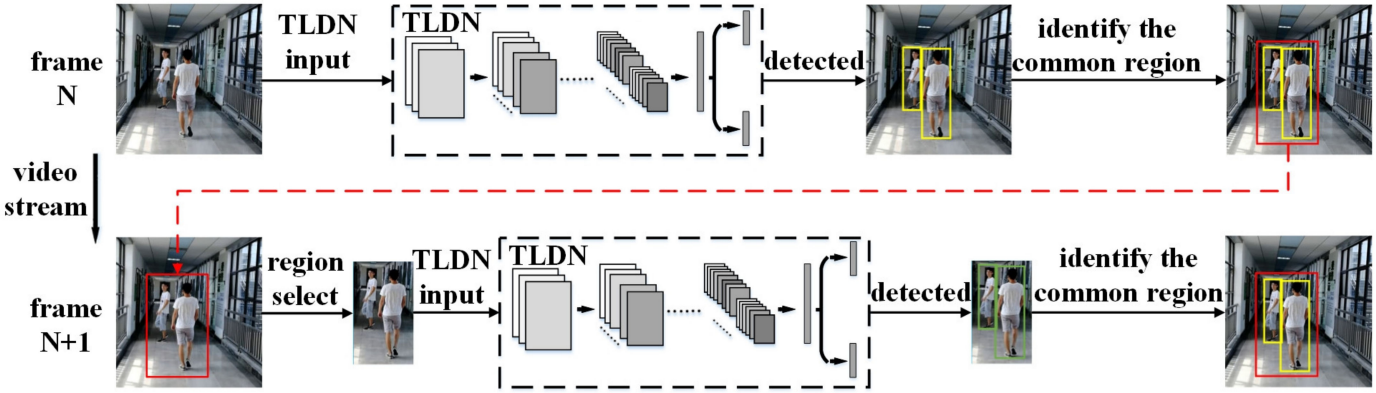

Figure 4. Proposed adaptive region focusing technique.

4. FPGA Design

The proposed pedestrian detection engine has been implemented on a Xilinx Zynq7020 FPGA, based on the deep learning processor unit (DPU). The DPU is an IP core provided by Xilinx for accelerating neural networks. It is able to support different neural network structures through reconfigurable hardware architecture (i.e., a neural network hardware accelerator including an array of processing engines, which are reused to execute the network layer-by-layer). As shown in Figure 5, the ZYNQ processing system based on the ARM Cortex A9 processor is used for realizing a Linux operating system, while the DPU is synthesized using FPGA logic as a hardware accelerator to communicate with the ZYNQ through AXI bus, for accelerating the neural network. During the operation, the main software program is executed partly in the Linux and partly on the DPU (the neural network part). The FPGA chip can communicate external peripherals, such as camera, PC, and SD card for data exchange. This forms a heterogeneous computing system for applications involving neural network computation.

The FPGA implementation flow is shown in Figure 6. First, the TLDN network model is quantized and compiled into DPU instruction codes by DNNDK, which is a toolchain provided by Xilinx. DPU is synthesized using FPGA logic as a hardware accelerator to communicate with the ZYNQ through AXI bus, for accelerating the neural network. Moreover, the inputs and outputs of the DPU unit are input nodes and output nodes of the TLDN network model. Then, a C++ program is written to initialize the DPU kernel, preprocess image, or video stream, before feeding them into the DPU task, processing on DPU, gaining the DPU output, and operating the postprocess on CPU. After completing the steps described above, the final result of the model will be gained. It also implements the algorithm to perform the ARFT operations, such as region identification and region selection. After that, the C++ program is cross-compiled together with the DPU instruction codes, which is included in the DPU driver provided by Xilinx, to generate an executable file to run in the Linux operating system. In the meantime, the root and kernel files for the ARM-based Linux operating system are built and placed in the SD card. During the

operation, the root and kernel files are loaded into ZYNQ to start up the Linux operating system. Then, the executable file is run to execute the C++ program partly in the Linux and partly on the DPU (the neural network part), which calls the DPU instruction codes for accelerating the TLDN when needed.

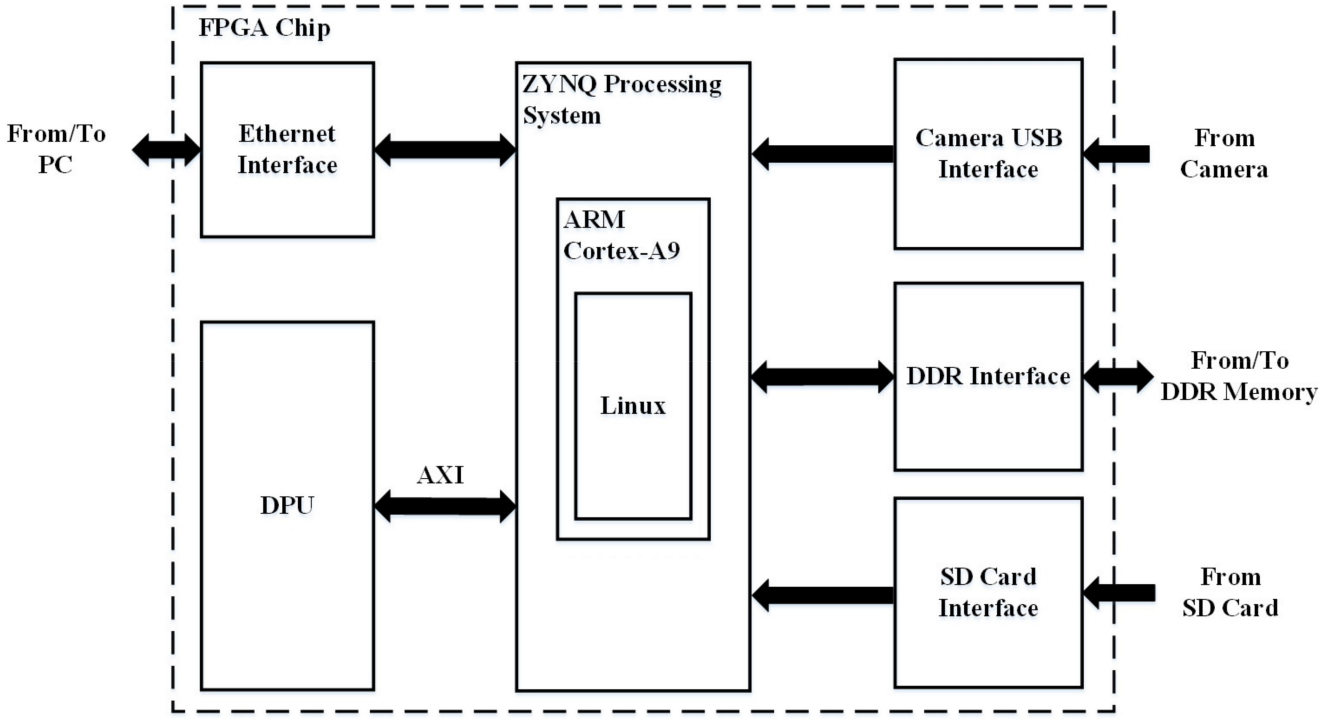

Figure 5. FPGA implementation.

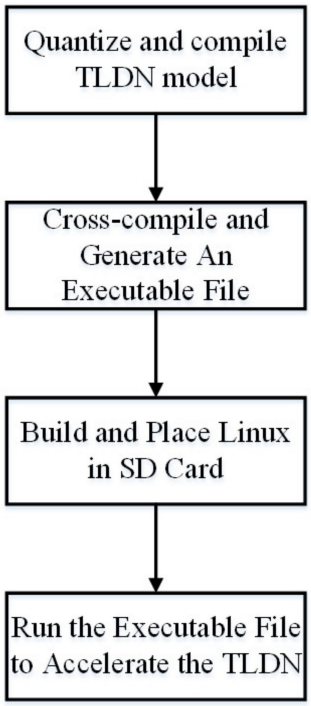

Figure 6. FPGA implementation flow.

Figure 7 shows the test setup. A PYNQ-Z2 board is used for testing the speed and power consumption, which includes the Xilinx Zynq7020 FPGA chip, USB port, DDR

memory, SD card, and Ethernet port. During the operation, the input video stream from a camera enters the system through the USB port. Based on the Linux system and OpenCv, the movie from the camera is read and preprocessed in the software before running the DPU task. The DPU implemented on the ZYNQ accelerates the TLDN layer-by-layer and exchanges intermediate feature map data with the on-board DDR. The pedestrian detection results are transferred to the PC through the Ethernet port, for monitoring. The architecture of the DPU is configured as B1152 (i.e., single DPU core), the system operating frequency is 140 MHz, and the resource utilization is shown in Table 2.

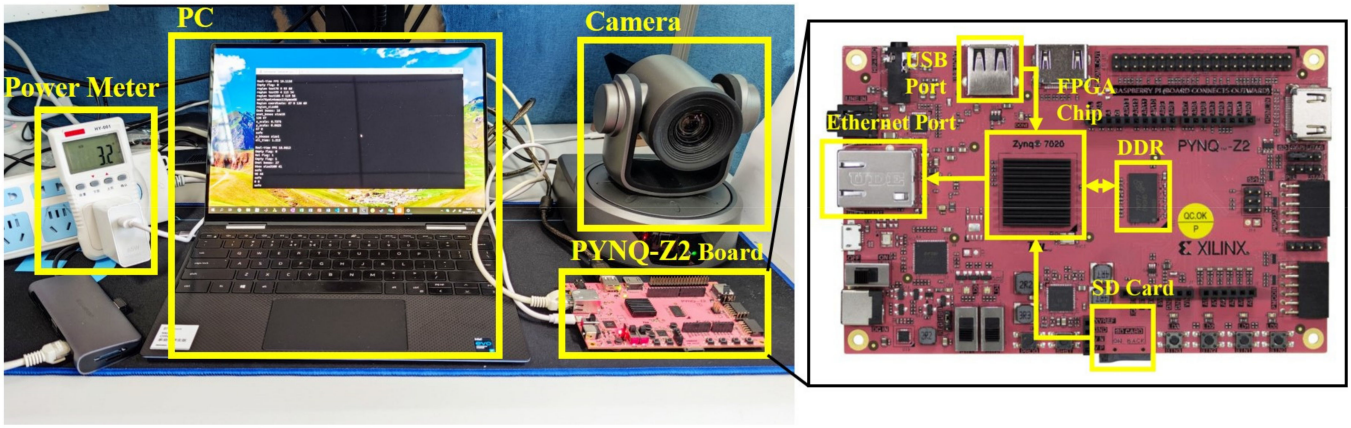

Figure 7. Experimental setup.

Table 2. Resource utilization.

	LUTs	Registers	DSPs	Block RAM
utilization	87,049	93,981	396	14,652 Kb

5. Experimental Results

Experiments have been conducted to evaluate the performance of the proposed pedestrian detection engine, and it has been compared with the existing pedestrian detection engines.

5.1. Training and Testing Dataset

The training dataset is built by combining a public dataset (i.e., Caltech dataset [27]) with a customized dataset. It includes more than 24,000 images from 110,000 pedestrians, under various scenes and illuminations. The training dataset is divided into the following three categories based on IoU: positive, partial, and negative, as shown in Table 3. The testing dataset is only the Caltech dataset, which includes 4024 images from 5051 pedestrians.

Table 3. Categories of training dataset.

Categories	IOU
Positive	$\geq$0.65
Partial	$\geq$0.4 $\cap$ <0.65
Negative	<0.3

Specifically, for the training, the mixture of the Caltech dataset and a customized dataset is used, which includes only static images. For testing the accuracy, the Caltech dataset is used, which includes only static images. For testing the speed and power consumption, a camera is used to capture 10 min of movie (30 fps) as the input of the proposed network. This is able to evaluate the effect of the proposed adaptive region focusing technique.

5.2. Training

The training data of the first-stage detection network are generated as follows. First, random bboxes, with constraints on the width (15 pixels to half of the image width) and height (39 pixels to half of the image height), are generated for each image. Then, IoU is calculated between the generated bboxes and the label bboxes, to obtain the categories in Table 3. During the training of the first-stage detection network, the images with generated bboxes are cropped in the original images and resized to 39×15. Then, they are sent to the first-stage detection network in batches for training. After the first-stage detection network, the 39×15 images become 1×1, which corresponds to a set of classification and border regression values. The training data of the second-stage detection network are generated in a similar way, except that the bboxes are generated by the first-stage detection network instead of random functions and the cropped images are resized to 104×40. The base learning rate of the two-stage network training is set to 0.01 and the optimizer is a momentum optimizer.

The detection network has the following two outputs: the pedestrian/non-pedestrian classification result and the bbox regression. Therefore, a joint loss function is adopted for the training as follows:

(1) Pedestrian classification: Pedestrian classification is used to distinguish whether the image in the frame is a pedestrian or a background, so this is a two-classification task. We use the cross-entropy loss function for training. For each sample x_i, we use the following function:

$$L_i^{cls} = -\left(y_i^{det} \log(p_i) + \left(1 - y_i^{det}\right)(1 - \log(p_i))\right) \tag{9}$$

where p_i is the network output for the sample x_i, which is used to indicate the probability that x_i is a pedestrian. $y_i^{det} \in \{0, 1\}$ is from the ground-truth tag and represents the true value.

(2) Frame regression: Frame regression is used to reduce the position gap between the real frame and the predicted frame. Each frame includes the following four pieces of information: left border, upper border, height, and width. Therefore, we adopt Euclidean distance measure loss, as follows:

$$L_i^{box} = \| \hat{y}_i^{box} - y_i^{box} \| \tag{10}$$

where $\hat{y}_i^{box}$ is the target frame obtained from the network output. y_i^{box} is the real coordinate information, and it includes four dimensions, so $y_i^{box} \in R^4$.

(3) Joint loss function: since the network needs to complete two different tasks at the same time, it cannot use (9) or (10) alone as the loss function, so the joint loss function is introduced as follows:

$$L_{sum} = \sum_{i=1}^{N} \left(\lambda_1 L_i^{cls} + \lambda_2 L_i^{box}\right) \tag{11}$$

where L_i^{cls} is the loss function for the pedestrian classification, which uses the cross-entropy function, L_i^{box} is the loss function for the bbox regression, which uses the Euclidean distance, and λ_1 and λ_2 are the weight coefficients for the two loss functions. For the first-stage detection network, we set $\lambda_1 = 1$ and $\lambda_2 = 0.5$. For the second-stage detection network, we set $\lambda_1 = 1$ and $\lambda_2 = 0.6$, to obtain more-accurate bbox coordinates.

5.3. Evaluation of Detection Accuracy

The detection accuracy is evaluated using *precision*, *recall*, *miss rate* and $false\ positives\ per\ image$, as shown in Equations (12)–(15). Figures 8 and 9 show the $precision - recall$ curve and *miss rate* curve of the proposed pedestrian detection engine. For comparison, we have evaluated some existing methods using the same testing dataset. As

shown in Figures 8 and 9, the proposed engine achieves a *precision* of 85.18% and *miss rate* of 25.16%, under the IoU threshold of 0.5, which are comparable to most of the existing methods.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{12}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{13}$$

$$miss\ rate = \frac{false\ negatives}{true\ positives + false\ positives} \tag{14}$$

$$false\ positives\ per\ image = \frac{false\ positives}{the\ number\ of\ image} \tag{15}$$

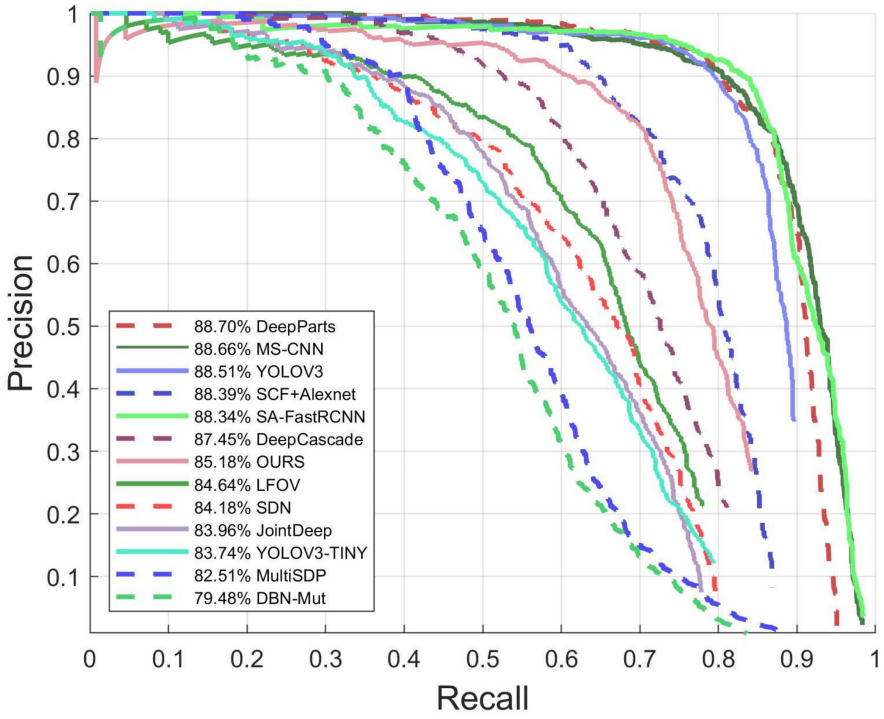

Figure 8. The precision–recall curve of the proposed method and the comparison with the existing methods.

5.4. Evaluation of Computational Complexity

Table 4 shows the computational complexity of the proposed engine and the comparison with the existing methods. Only the engines based on deep learning are listed here, as it is difficult to calculate the computational complexity of the engines based on conventional machine learning methods. The numbers of parameters and operations of deep learning-based engines are obtained by calculation using the neural network models provided in their papers. In Table 4, Ours-CR and Ours-SR both refer to the result of using a common region for the adaptive region focusing. It can be observed that our engine (Ours-CR) only requires 0.73 M parameters and 1.04 B operations. It outperforms the other engines in terms of the computational complexity, while achieving a comparable precision and miss rate. Compared to Ours-SR, Ours-CR has a reduced number of operations, due to the reduced number of computations of the detection network.

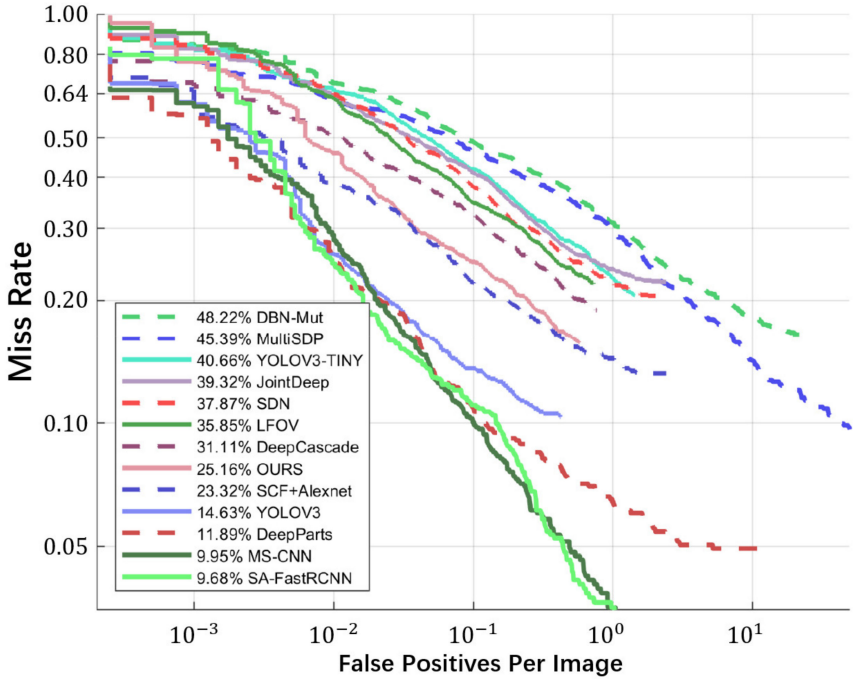

Figure 9. The miss rate curve of the proposed method and the comparison with the existing methods.

Table 4. Computation complexity comparison of different methods.

Methods	Parameters	Operations	Precision	Miss Rate
OURS-CR	0.73M	1.04B	85.18%	25.16%
OURS-SR	0.73M	2.75B	85.18%	25.16%
SDN [28] [2]	\	\	84.18%	37.87%
DeepParts [29] [2]	187.10M	6.81B	88.70%	11.89%
LFOV [30] [2]	135.27M	0.64B	84.64%	35.85%
SA-FastRCNN [31] [2]	266.84M	41.35B	88.39%	9.68%
MS-CNN [7] [2]	~217M	\	88.66%	9.95%
OR-CNN [32] [2]	138.34M	30.94B	\	4.1%
ALFNet [33] [2]	48.4M	5.07B	\	22.5%
CSP [34] [2]	~31.23M	~67.03B	\	4.5%
CSANet [35] [2]	~22.66M	~11.54B	\	3.88%
YOLOv3-Tiny [3]	7.86M	5.56B	83.74% [1]	40.66% [1]
YOLOv3 [36] [3]	61.57M	65.86B	88.51% [1]	14.63% [1]
YOLOV4 [37] [3]	64.03M	62.25B	88.66% [1]	10.21% [1]
YOLOV5s [3]	7.3M	17.0B	88.16% [1]	25.65% [1]
MDFL [38] [2]	~276.69	~61.88B	\	31.46%
MultiSDP [39] [2]	\	\	82.51%	45.39%
DBN-Mut [40] [2]	\	\	79.48%	48.22%
SCF+AlexNet [41] [2]	233M	727M	88.39%	23.32%

[1] This accuracy is obtained by replication the work and testing it using the same testing dataset. [2] Pedestrian detection network. [3] Objection detection network.

5.5. Evaluation of Speed and Power Consumption

Table 5 shows the measured power consumption and frame rate per second of the proposed engine for FPGA implementation. For comparison, we have also implemented YOLOv3 and YOLOv3-Tiny on the same FPGA board. However, DPU does not support the upsampling structure in YOLOv4 and v5, which is why we only implemented YOLOv3 and YOLOv3-Tiny for comparison. The power consumption is measured by using a power meter on the electrical socket of the FPGA board, as shown in Figure 7. As the FPGA board has a constant power consumption of ~2.6 W without any design implementation, this power consumption is removed during the measurement, for all the compared methods. It

can be observed that the power consumption of the proposed engine achieves 16.3 FPS, while consuming 0.58 W on the PYNQ-Z2 FPGA board, which is significantly better than that of the YOLOv3 (5.3 Fps, 2.43 W) and YOLOv3-Tiny (12.8 Fps, 0.95 W).

Table 5. FPS and power of different methods [1].

Methods	FPS	Power (W)
OURS-CR	16.3	0.59
OURS-SR	8.6	0.68
YOLOv3-Tiny	12.8	0.95
YOLOv3 [36]	5.3	2.43

[1] The input image size is 224 × 224.

6. Conclusions

In this paper, we propose a lightweight pedestrian detection engine with a two-stage low-complexity detection network and an adaptive region focusing technique, to reduce the computational complexity in pedestrian detection, while maintaining a high detection accuracy. Compared to the existing designs, the proposed pedestrian detection engine significantly reduces the number of parameters and operations, with a comparable precision (85.18%) and miss rate (25.16%). The proposed design has been implemented on FPGA for the evaluation of its real-time performance and power consumption. It is able to achieve 16.3 Fps while consuming 0.59 W, which is better than the mainstream detection engines, such as YOLOv3 (5.3 Fps, 2.43 W) and YOLOv3-Tiny (12.8 Fps, 0.95 W).

Author Contributions: Conceptualization, J.Z., L.Q. and T.Z.; methodology, J.Z., L.Q. and T.Z.; software, L.Q., T.Z. and H.G.; validation, L.Q. and T.Z.; formal analysis, J.Z., L.Q., T.Z. and L.C.; investigation, L.Q., T.Z., C.J. and Y.G.; resources, L.Q., T.Z., C.J. and Y.G.; data curation, L.Q. and T.Z.; writing—original draft preparation, L.Q. and T.Z.; writing—review and editing, J.Z., L.Q., H.G. and L.C.; visualization, L.Q. and T.Z.; supervision, J.Z.; project administration, L.Q.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by NSAF (grant No. U2030204).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the In IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005; pp. 886–893.
2. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef] [PubMed]
3. Wojek, C.; Schiele, B. A performance evaluation of single and multi-feature people detection. In *Pattern Recognition*; Lecture Notes in Computer Science; Rigoll, G., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5096, pp. 82–91.
4. Dollar, P.; Appel, R.; Belongie, S.; Perona, P. Fast Feature Pyramids for Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1532–1545. [CrossRef] [PubMed]
5. He, Y.; Qin, Q.; Vychodil, J. A Pedestrian Detection Method Using SVM and CNN Multistage Classification. *J. Inf. Hiding Multim. Signal Process.* **2018**, *9*, 51–60.
6. Zhang, X.; Cheng, L.; Li, B.; Hu, H. Too Far to See? Not Really!—Pedestrian Detection With Scale-Aware Localization Policy. *IEEE Trans. Image Process.* **2018**, *27*, 3703–3715. [CrossRef]
7. Cai, Z.; Fan, Q.; Feris, R.S.; Vasconcelos, N. A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; Volume 9908, pp. 354–370.
8. Lin, C.; Lu, J.; Wang, G.; Zhou, J. Graininess-Aware Deep Feature Learning for Pedestrian Detection. In *European Conference on Computer Vision*; Lecture Notes in Computer Science; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11213, pp. 745–761.
9. Gao, Z.; Li, S.; Chen, J.; Li, Z. Pedestrian Detection Method Based on YOLO Network. *Comput. Eng.* **2018**, *44*, 215–219. [CrossRef]

10. Peng, Q.; Luo, W.; Hong, G.; Feng, M.; Xia, Y.; Yu, L.; Hao, X.; Wang, X.; Li, M. Pedestrian Detection for Transformer Substation Based on Gaussian Mixture Model and YOLO. In Proceedings of the 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics, Hangzhou, China, 27–28 August 2016; pp. 562–565.
11. Byeon, Y.-H.; Kwak, K.-C. A Performance Comparison of Pedestrian Detection Using Faster RCNN and ACF. In Proceedings of the 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Hamamatsu, Japan, 9–13 July 2017; pp. 858–863.
12. Xiaoqian, Y.; Yujuan, S.; Liangliang, L. Pedestrian detection based on improved Faster RCNN algorithm. In Proceedings of the 2019 IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 11–13 August 2019; pp. 346–351. [CrossRef]
13. Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
14. Wang, X.; Han, T.X.; Yan, S. An HOG-LBP Human Detector with Partial Occlusion Handling. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 32–39.
15. Sabzmeydani, P.; Mori, G. Detecting pedestrians by learning shapelet features. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1328–1335.
16. Schwartz, W.R.; Kembhavi, A.; Harwood, D.; Davis, L.S. Human Detection Using Partial Least Squares Analysis. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 24–31.
17. Wang, S.; Cheng, J.; Liu, H.; Wang, F.; Zhou, H. Pedestrian Detection via Body Part Semantic and Contextual Information With DNN. *IEEE Trans. Multimed.* **2018**, *20*, 3148–3159. [CrossRef]
18. Liu, T.; Stathaki, T. Enhanced Pedestrian Detection using Deep Learning based Semantic Image Segmentation. In Proceedings of the 2017 22nd International Conference on Digital Signal Processing, London, UK, 23–25 August 2017.
19. Lin, C.-Y.; Xie, H.-X.; Zheng, H. PedJointNet: Joint Head-Shoulder and Full body Deep Network for Pedestrian Detection. *IEEE Access* **2019**, *7*, 47687–47697. [CrossRef]
20. Cao, J.; Pang, Y.; Han, J.; Gao, B.; Li, X. Taking a Look at Small-Scale Pedestrians and Occluded Pedestrians. *IEEE Trans. Image Process.* **2020**, *29*, 3143–3152. [CrossRef] [PubMed]
21. Yin, R. Multi-resolution generative adversarial networks for tiny-scale pedestrian detection. In Proceedings of the 2019 IEEE International Conference on Image Processing, Taipei, Taiwan, 22–25 September 2019; pp. 1665–1669.
22. Li, X.; Liu, Y.; Chen, Z.; Zhou, J.; Wu, Y. Fused discriminative metric learning for low resolution pedestrian detection. In Proceedings of the 2018 25th IEEE International Conference on Image Processing, Athens, Greece, 7–10 October 2018; pp. 958–962.
23. Kruthiventi, S.S.S.; Sahay, P.; Biswal, R. Low-light pedestrian detection from rgb images using multi-modal knowledge distillation. In Proceedings of the 2017 24th IEEE International Conference on Image Processing, Beijing, China, 17–20 September 2017; pp. 4207–4211.
24. Xu, D.; Ouyang, W.; Ricci, E.; Wang, X.; Sebe, N. Learning Cross-Modal Deep Representations for Robust Pedestrian Detection. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4236–4244.
25. Chuanyi, H.; Jinlei, Z.; Feng, L.; Shengkai, W.; Houjin, C. Design of lightweight pedestrian detection network in railway scenes. *J. Phys. Conf. Ser.* **2020**, *1544*, 012053. [CrossRef]
26. Huang, R.; Pedoeem, J.; Chen, C. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In Proceedings of the 2018 IEEE International Conference on Big Data, Seattle, WA, USA, 10–13 December 2018; pp. 2503–2510.
27. Dollar, P.; Wojek, C.; Schiele, B.; Perona, P. Pedestrian Detection: A Benchmark. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; Volumes 1–4; pp. 304–311.
28. Luo, P.; Tian, Y.; Wang, X.; Tang, X. Switchable Deep Network for Pedestrian Detection. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 899–906.
29. Tian, Y.; Luo, P.; Wang, X.; Tang, X. Deep Learning Strong Parts for Pedestrian Detection. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1904–1912.
30. Angelova, A.; Krizhevsky, A.; Vanhoucke, V. Pedestrian Detection with a Large-Field-Of-View Deep Network. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 704–711.
31. Li, J.; Liang, X.; Shen, S.; Xu, T.; Feng, J.; Yan, S. Scale-Aware Fast R-CNN for Pedestrian Detection. *IEEE Trans. Multimed.* **2018**, *20*, 985–996. [CrossRef]
32. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; Volume 11207, pp. 657–674.
33. Liu, W.; Liao, S.; Hu, W.; Liang, X.; Chen, X. Learning Efficient Single-Stage Pedestrian Detectors by Asymptotic Localization Fitting. In Proceedings of the Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; Volume 11218, pp. 643–659.
34. Liu, W.; Liao, S.; Ren, W.; Hu, W.; Yu, Y.; Soc, I.C. High-level Semantic Feature Detection: A New Perspective for Pedestrian Detection. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 5182–5191.
35. Zhang, Y.; Yi, P.; Zhou, D.; Yang, X.; Yang, D.; Zhang, Q.; Wei, X. CSANet: Channel and Spatial Mixed Attention CNN for Pedestrian Detection. *IEEE Access* **2020**, *8*, 76243–76252. [CrossRef]

36. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
37. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
38. Lin, C.; Lu, J.; Zhou, J. Multi-Grained Deep Feature Learning for Robust Pedestrian Detection. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 3608–3621. [CrossRef]
39. Zeng, X.; Ouyang, W.; Wang, X. Multi-stage Contextual Deep Learning for Pedestrian Detection. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 121–128.
40. Paisitkriangkrai, S.; Shen, C.; van den Hengel, A. Strengthening the Effectiveness of Pedestrian Detection with Spatially Pooled Features. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 546–561.
41. Hosang, J.; Omran, M.; Benenson, R.; Schiele, B. Taking a deeper look at pedestrians. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 4073–4082.