A CPU-based Pedestrian Detector using Deep Learning for Intelligent Surveillance Systems

Muhamad Dwisnanto Putro, Duy-Linh Nguyen and Kang-Hyun Jo
Department of Electrical, Electronic, and Computer Engineering, University of Ulsan
Ulsan, Korea
Email: dputro@mail.ulsan.ac.kr; ndlinh301@mail.ulsan.ac.kr; acejo@ulsan.ac.kr

***Abstract*—This paper presents real-time pedestrian detection based on a deep learning approach for localizing human areas. It introduces lightweight architecture using a convolutional neural network that can smoothly work on a CPU device. Specifically, the framework contains three main modules. Firstly, light extractor features utilize an Efficient Bottleneck Partitioning (EBP) block to distinguish person features and background features rapidly. Secondly, the Path Aggregation Network (PANet) helps the network's ability to obtain fused information from different feature frequencies. Lastly, Triple prediction anchor-based layers to predict various human sizes. An end-to-end network is comprehensively trained on the MS COCO 2017 dataset to gain knowledge of discrimination for multiple challenges. The pedestrian detection network (PdNet) produces lower trainable parameters than other detectors. As a result, it achieves competitive performance evaluated on the MS COCO 2017, PASCAL VOC 2007, and 2012 datasets. Moreover, the main advantage of this detector is that it can run 14 frames per second when working in real-time on CPU devices.**

***Index Terms*—Pedestrian detector, convolutional neural network, CPU device, real-time.**

I. Introduction

Several applications utilize pedestrian detection in broad aspects, such as self-driving, person re-identification, pose estimation, human action recognition, and robotics. Pedestrian is also an essential object of a video surveillance system, so this detection has become a trending task in the field of computer vision. It localizes regions of the entire person body in an image by providing bounding boxes as predicted areas [1]. Intelligent systems have developed rapidly by employing video surveillance to prevent criminal acts. The monitoring system requires this detector to analyze video content sourced from the camera [2]. In addition, the information can respond periodically to identify abnormal behavior. Therefore, video surveillance plays an important task for security systems to monitor human activities in indoor and outdoor areas. Person detection is a fundamental process for observing pedestrian behavior systems. Video surveillance is required to work all the time and under different illumination conditions [3]. It is a challenge that is necessary addressed to evaluate the robustness of a detector. Besides, the model's efficiency supports a method to be feasible to be implemented in low-cost devices [4].

The previous studies have produced pedestrian detectors using conventional methods [5]–[7]. Zhang et al. detected three main human components: the head, upper body, and lower body [5]. It employs statistical models and a pool of rectangular features for variations in clothing or environmental settings. Additionally, Haar-like features produce the different characteristics between parts of the human body. Chen et al. applied traditional feature descriptors and feature selection methods to identify human body areas [6]. This work also proposed a novel Oriented Chamfer Distance (OCD) feature to distinguish pedestrian and background. On the other hand, a low-cost pedestrian detection has been introduced using a reliable cascade classifier [7]. It uses Viola and Jones extractor features to obtain distinctive components. The methods above have much potential practical value that can work in real-time on a device. However, those methods are weak in detecting a person in multi-appearance, multi-pose, angle of view, occlusion, and illuminance challenges. The feature extraction methods used are difficult to discriminate against complex features from the background.

In recent years, the Convolutional Neural Network (CNN) has been widely used as robust feature extraction. This approach is powerful for recognizing specific features of pedestrians [8]–[12]. Faster RCNN [13], SSD [14], and YOLOV3 [15] dominate the human body localization work. These architectures produce high computing power, so it tends to be expensive. It depends on graphic accelerators. In addition, these detectors obtain slow processing speed when implemented in Central Processing Unit (CPU) devices. On the other hand, a lightweight CNN-based architecture runs smoothly on the CPU [16]. This structure requires the suitability of light convolutional layers to achieve competitive performance with benchmark architectures. The trade-off between speed and accuracy emphasizes the balance of the CNN model to fast and robust work in real-world scenarios [17]. Therefore, the proposed detector is designed to run smoothly on CPU devices without compromising performance.

A pedestrian proposed network (PdNet) introduces a light backbone that utilizes an Efficient Bottleneck Partitioning (EBP) module to quickly discriminate against specific features of the human body. It separates the two parts of the feature map using simple convolution to reduce the constraints of high computation and significant parameters. A Path Aggregation Network [18] helps networks get information from multiple dimensions and feature levels. It also supports multi-level

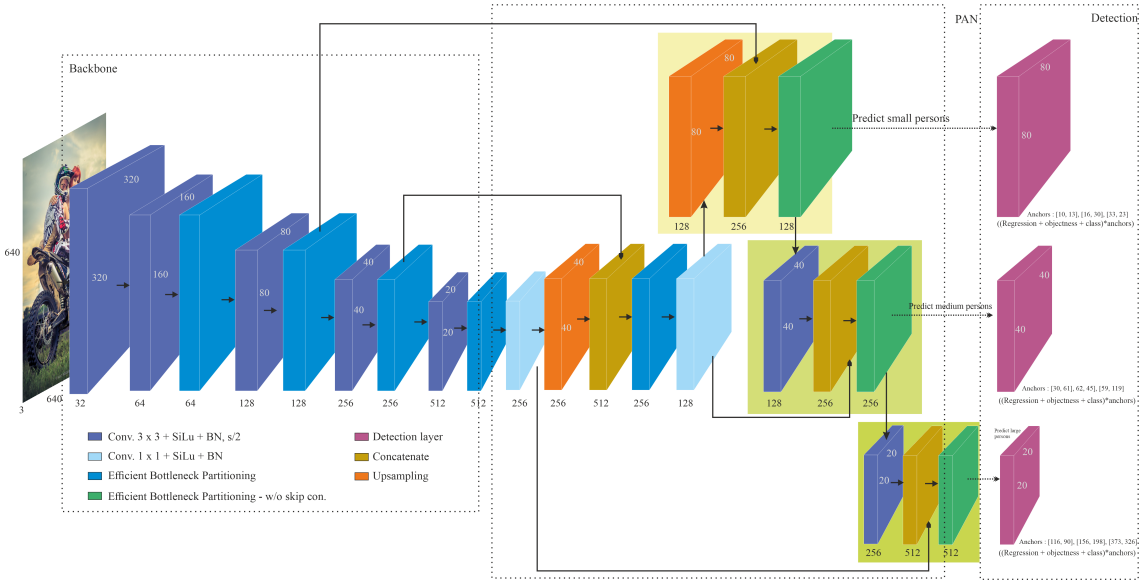

Fig. 1. The proposed architecture. A backbone module is used to extract human body features with efficient bottleneck partitioning module. Besides, the PANet and detection modules help the detector to identify the location of pedestrian in multi-scale variant.

detection to predict various human body sizes by assigning different anchor sizes [19]. The efficient architecture emphasizes the efficiency model to produce a lightweight detector that can run smoothly on low-cost devices. It also eliminates the dependence on expensive hardware from the CNN model, which uses a graphic accelerator in inference stages. Based on this description, the main contributions of the work are summarized as follows:

1) A novel real-time pedestrian detection (PdNet) is proposed to locate an entire human body quickly and accurately that can run smoothly on a CPU device to be implemented in video surveillance.
2) A bottleneck module is improved to be more efficient. It supports the detector to work fast without compromising its accuracy. As a result, it achieves competitive performance from other architecture on MS COCO 2017 [20] and PASCAL VOC [21].

II. Proposed Architecture

The proposed architecture contains three main modules, as shown in Fig. 1. First, a backbone as a baseline module to comprehensively extract human body features with Efficient Bottleneck Partitioning (EBP). Secondly, the PANet (Path Aggregation Network) connect the relationship of the different multi-frequency feature map. Lastly, the detection module localizes the person area using a multi-scale region. The EBP module is used in the backbone and PANet to discriminate against human body features efficiently.

A. Backbone Module

The backbone module is assigned to comprehensively extract the input features map that involves a series of convolution operations. In addition, it also reduces the feature map size to save computing power while providing abundant object

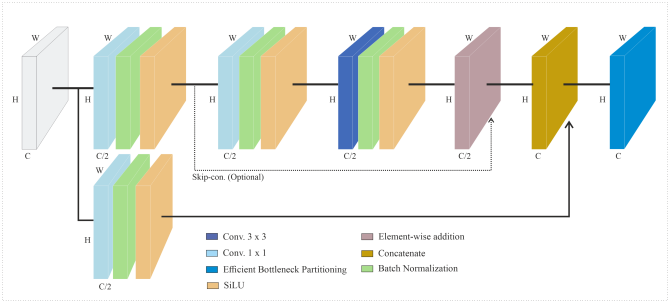

Fig. 2. An efficient bottleneck partitioning fast discriminates the interest features.

features through multi-layer channels. The proposed detector applies 3×3 convolution with strides of two to gradually shrink the feature map. Therefore, the last feature map of the backbone module is reduced by 32 times from its original size. On the other hand, the number of channels from each feature map increases and produces 512 at the end of the feature map. The SiLU activation and batch normalization follow the whole convolution operation to prevent vanishing and exploding gradient in each neuron. Several works [14], [15] have used 3×3 Convolution as a robust filter to distinguish the target object against the background. They employ this block according to their respective demands.

Furthermore, the proposed detector introduces efficient bottleneck partitioning to extract the reduced size feature map comprehensively. This approach increases the efficiency of the original bottleneck block [22] by dividing the input feature map into two partitions, as shown in Fig. 2. The channel of the input feature x_i is split in half for each part by employing

a 1 × 1 convolution, illustrated as:

$$x_i = [F_1(x_i),\ F_2(x_i)], \tag{1}$$

where F_1 and F_2 are convolution operations using 1 × 1 kernel with SiLU activation (S) and batch normalization (Bn). It then applies the residual bottleneck module to only one partition and fuses both partitions at the end of the module, defined as:

$$y_i = F_1(x_i) + S(Bn(W_2 S(Bn(W_1 F_1(x_i) + b_1)) + b_2)), \tag{2}$$

$$z_i = F_2(x_i). \tag{3}$$

A bottleneck module employs sequentially 1 × 1 and 3 × 3 convolution to extract essential features. However, the proposed module does not reduce the number of channel layers at the beginning of the convolution to maintain the quantity of information from the feature map. Finally, the output of this module combines the feature map at the end of the module by fusing the extracted features y_i and the identity of the other partitions z_i, described as:

$$EBP_i = [y_i,\ z_i]. \tag{4}$$

The whole convolution operation is also followed SiLU and batch normalization. EBP modules are applied to the second to fifth stages with the number of modules at each stage, including 1, 3, 3 and 3, respectively.

Bottleneck extracts essential features by providing a variety of filter types and applying them sequentially. It informatively preserves important features using residual functions to correct the feature map. On the other hand, EBP emphasizes on saving of parameters and computational. It focuses on generating small channels at the beginning of the module, so the bottleneck is operating the kernel with less computation. However, to prevent information loss due to channel shrinkage, the identity module is applied to another partition and fuses it with the extracted bottleneck. Therefore, the output of this module generates computational efficiency while maintaining feature extraction performance without significantly losing feature information.

B. Triple Stages of Path Aggregation Network

A CNN architecture generates a different level of feature map at each layer. The initial layer produces low-level features representing simple shapes, while the forward layer provides mid and high-level features that describe complex textures. Feature information from this level is essential to be processed and used in making prediction decisions. The pyramid feature [18] connects the layers with different frequencies, thereby helping to improve the performance of the CNN model. The PdNet detector applies three-stage blocks with different feature map sizes, including 80, 60, and 20. It uses upsampling at the end of the backbone feature map, then concatenates with the feature map at the previous level with the same spatial size. An EBP is applied to extract features at the beginning of the stage efficiently. It helps the detector acquire more specific human body features. In addition, 3 × 3 convolution is also applied to reduce feature maps used to generate feature maps with different scales for multi-stage prediction assignments.

C. Detection Module

Object detection uses the detection layer to predict the coordinates of (x, y, h, w) and the object's class. Faster RCNN [13] applies a one-stage detection layer to predict objects of various sizes. This structure imposes to detect different scales in one spatial dimension, limiting a particular scale to be identified. Therefore, the PdNet detector applies multi-scale detection layers to accommodate variations in object size. Variations in the size of the feature map correspond to different sizes of person objects. This method helps the detector to focus more on employing each layer to predict objects at specific scales. The feature map is 80 for predicting small people, 40 for middle people, and 20 for large scale. Each feature map employs three anchors of varying sizes, as shown in Fig. 1. In addition, EBP is also applied on the previous detection layer to improve specific features before the number of channels is generated according to the number of anchors and classes. Instead of using residuals on bottleneck blocks, this ignores them to reduce the computation of the module.

D. Loss Function

Generally, neural networks utilize the loss function to measure the inaccuracy between the predicted scores and ground truth. Therefore, it encourages neural weights to optimally work so it can minimize errors. The PdNet detector localizes the area of the suspected human body by predicting the coordinates and size of the box. It also estimates the objectness and probability of class (pedestrian and none). Each detection layer applies 3 × 3 to generate both offsets, adjusting the number of channels based on the number of anchors. The PdNet detector uses three losses: regression loss, objectness loss, and classification loss to calculate prediction error of the bounding box location, object or no object, and classification error. It applies to each i grid and j anchor box as expressed as follows.

$$\begin{aligned} Loss = \lambda_{box} \sum_{i=1}^{s^2} \sum_{j=1}^{B} \mathbf{1}_{ij}^{obj} L_{box} + \lambda_{obj} \sum_{i=1}^{s^2} \sum_{j=1}^{B} \mathbf{1}_{ij}^{obj} L_{obj} \\ + \lambda_{cls} \sum_{i=1}^{s^2} \sum_{j=1}^{B} \mathbf{1}_{ij}^{obj} L_{cls}, \end{aligned} \tag{5}$$

where B is number of anchors, s^2 is grid area. 1_{ij}^{obj} is equal to one when there is an object in the cell, and 0 otherwise. It sets $\lambda_{box} = 0.05, \lambda_{obj} = 1$, and $\lambda_{cls} = 0.5$ as balancing parameters in regression, objectness, and classification loss, respectively. These parameters adopt a work [23]. L_{box} applies complete IoU loss [24] which accumulates the difference of IoU area, distance between box points, and consistency of aspect ratio comparison. L_{obj} uses confidence loss [19], while L_{cls} uses binary cross-entropy [19].

III. Implementation Details

The PdNET implementation is built on the PyTorch framework. The model is conducted with GTX 1080Ti as a GPU accelerator. Then the trained model is tested on Intel Core

TABLE I
ABLATION STUDY. AVERAGE PRECISION (AP) ON MS COCO DATASET.

Model	Number of parameters	AP (%)
Bottleneck Wihout EBP	9,770,518	0.528
Bottleneck With EBP	**6,954,838**	**0.524**

I5-6600 CPU @ 3.30GHz, 32GB RAM. The robust model is trained on MS COCO 2017 that contains 117,266 images. To enrich the variety of data, it applies augmentation: color distortion, vertical and horizontal flipping. Then random cropping is used, and insert the augmented images on the mosaic frame. Finally, it resizes the mosaic images to 640 $\times$ 640. In the training stage, the Stochastic Gradient Descent is an optimizing method to update neural weights with 10^{-2} is the initial learning rates and update by $2 \cdot 10^{-1}$ in the final OneCycleLR learning rate. The weight decay is $5 \cdot 10^{-4}$, and the momentum is 0.937. It uses batch sizes of 32 for the entire training dataset. The IoU (Intersection over Union) threshold in training stages is 0.5 to generate the best bounding box.

IV. EXPERIMENTAL RESULTS

In this section, the PdNet architecture is evaluated on MS COCO 2017, PASCAL VOC 2007, and PASCAL VOC 2012 datasets. Besides, another experiment shows the PdNet speed was tested on CPU and compares to other detectors.

A. Ablative Study

The PdNet detector employs EBP as an efficient feature extractor. This module plays an essential role in supporting the network to operate in real-time without compromising accuracy. Table I shows that EBP can emphasize the number of parameters of the proposed detector. This experiment shows a difference of 2.8M parameters with the use of a common bottleneck. A bottleneck without EBP removes the efficient partitioning module at all layers, including backbone, PANet, and detection, so it is the same as the vanilla bottleneck module. On the other hand, the accuracy showed a non-significant difference between the two experiments when evaluated on the MS COCO dataset with the primary challenge metric.

B. Evaluation on Datasets

1) MS COCO 2017: This dataset contains a lot of instances with complex challenges with different poses, the object scale, and the occlusions. Therefore, the proposed detector can learn the object, and excellent generalizes with complicated environmental conditions. The MS COCO 2017 dataset generally consists of 122,218 labeled images and 80 object classes. It divides 118k train for and 5k validation. Furthermore, it uses 66,808 person images extracted from this dataset for pedestrian detection. In addition, we chose 64,115 human images for the training processing, and the rest were used for validation. Table II shows that the PdNet achieved 52.40% AP on the primary metric (IoU=.50:.05:.95). This performance is slightly lower by 0.8% AP with the YOLOV5 small version. On the other hand, the YOLOV3 tiny version is higher than PdNet's performance and differs by 1.25% AP using Darknet19 as the backbone module. However, the proposed detector can detect a small person in indoor and outdoor conditions, as shown in Fig. 3 (a).

TABLE II
EVALUATION RESULTS ON MS COCO 2017, PASCAL VOC 2007 AND PASCAL VOC 2012 DATASETS.

Model	AP (%)	Backbone	Training dataset
Evaluation on MS COCO 2017			
ResNet18 (0.25)	40.10	ResNet18	COCO17
ShuffleNetv2(0.5)	33.80	ShuffleNetv2	COCO17
MobileNetV2(0.33)	39.10	MobileNetV2	COCO17
PeleeNet(0.5)	41.90	PeleeNet	COCO17
Bai et al. [10]	45.50	Manually designed	COCO17
Tiny model-Bai et al. [12]	42.90	Manually designed	COCO17
Tiny-YOLOv2 [15]	44.97	Darknet19	COCO17
Tiny-YOLOv3 [15]	53.65	Darknet19	COCO17
YOLOV5-small [23]	53.20	CSPBottleneck	COCO17
PdNet	**52.40**	**Manually designed**	**COCO17**
Evaluation on PASCAL VOC 2007			
Improved Faster_RCNN [13]	75.65	VGG16	Caltech
TinyYOLOV2	63.88	Darknet19	VOC07
Tiny-YOLOV3	68.54	Darknet19	VOC07
Improved Tiny-YOLOv3 [11]	73.98	Darknet19	VOC07
Enhanced Tiny-Yolov3 [11]	78.64	Darknet19	VOC07
YOLOV5-small [23]	88.80	CSPBottleneck	COCO17
PdNet	**88.30**	**Manually designed**	**COCO17**
Evaluation on PASCAL VOC 2012			
Faster R-CNN [8]	62.90	VGG16	VOC07++VOC12
SSD512 [8]	39.40	VGG16	VOC07++VOC12+ COCOTRAINVAL35K
RefinedDet320	58.50	VGG16	VOC07++VOC12+ COCOTRAINVAL35K
RefinedDet320+	61.60	VGG16	VOC07++VOC12+ COCOTRAINVAL35K
RefinedDet512	63.60	VGG16	VOC07++VOC12+ COCOTRAINVAL35K
RefinedDet512+	66.00	VGG16	VOC07++VOC12+ COCOTRAINVAL35K
RFBNet300	29.00	VGG16	COCOTRAINVAL35K
RFBNet512-E	32.60	VGG16	COCOTRAINVAL35K
RFBMobileNet	23.80	MobileNet	COCOTRAINVAL35K
YOLO-AF-MS [8]	77.30	CSPDarkNet53	COCOTRAINVAL35K
YOLOV5-small [23]	88.90	CSPBottleneck	COCO17
PdNet	**88.40**	**Manually designed**	**COCO17**

2) PASCAL VOC 2007: This dataset contains around 9,963 labeled images with 20 object classes, described as follows: Person, Bird, Cat, Cow, Dog, Horse, Sheep, Airplane, Bicycle, Boat, Bus, Car, Motorbike, Train, Bottle, Chair, Dining table, Potted plant, Sofa, and TV/Monitor. It provides for image classification and detection. This dataset also contains more complex backgrounds, varying human postures, and occlusion. It only uses 2,007 images for evaluation of the proposed detector that only includes the human object. Table II shows that PdNet achieves an AP of 88.30% that is 0.5% different from the YOLOV5 small version as the leading competitor. The insignificant difference is not bad because the qualitative results show the proposed detector can detect the human body in different scales and occlusions, as shown in Fig. 3 (b).

3) PASCAL VOC 2012: This dataset contains 16,135 images, of which 5,138 images have ground truth with 20 types of class. The categories are the same as PASCAL VOC 2007. However, it only uses person class to evaluate the proposed

Fig. 3. Qualitative results from the prediction of the PdNet detector on the MS COCO 2017 (a), PASCAL VOC 2007 (b), PASCAL VOC 2012 (c), RGB video on VGA resolution (d), and infrared video on HD resolution (e).

detector. A total of 2,093 images with ground truth is used to examine pedestrian detectors. The PdNet achieves an AP of 88.40% in this evaluation dataset. It differs 0.50% with YOLOV5 small architecture, as shown in Table II. However, the critical difference contrasts with the number of parameters produced by each detector which differs by 1M parameters.

C. Runtime Performance on CPU

In general, CNN benchmark architectures generate a lot of computation and parameters because it uses many operations to compute the weighted layer. Therefore, they need expensive hardware to operate the model in real-time applications because the detectors are very slow to work using CPU-based devices. The graphic accelerator must be used to be implemented for practical applications. The proposed detector generates 6,954,838 parameters with 11.9 GFLOPS. These results encourage PdNet to operate in real-time on Intel Core I5-6600 CPU @ 3.30 GHz. Faster RCNN and SSD are very slow on this device. These detectors are not feasible to run on this CPU device, so they are irrelevant in real application scenarios. Fig. 4 shows that the proposed detector achieves 11.9 FPS, faster than other competitors. Although the precision is slightly below the YOLOV5 small version, this competitor is 4 FPS slower. Fig. 3 (c) and (d) show that the proposed detector works well when implemented as a pedestrian detector for video surveillance. It is robust in localizing the human body on low-illuminance conditions with infrared-based cameras. There are occluded persons by other objects, but it does not obstruct the PdNet from precisely localizing the human body area. It is because the proposed detector comprehensively learns specific human body features from complex instances and complicated environmental conditions. Although the dataset only provides images of the person in general, the person and pedestrian features are the same. It can help the proposed architecture create an accurate pedestrian

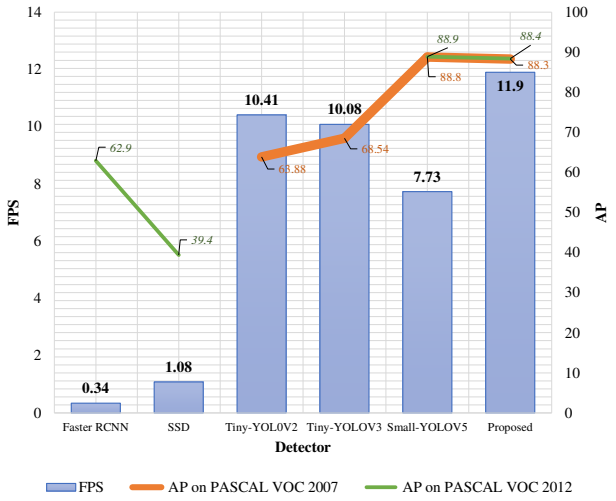

Fig. 4. Comparison of detector data processing speeds on a CPU in VGA resolution

detector. Additionally, the learning-based detector emphasizes the efficiency of computation power and can operate in real-time on low-cost devices.

V. Conclusion

This paper presents a CNN-based pedestrian detector for the video surveillance system. This detector aims to be able to operate in real-time without the accelerator graphic. The PdNet consists of a backbone, PANet, and detection module. An efficient bottleneck partitioning plays an essential role in reducing computational overhead. In addition, this module also helps the detector to produce fewer parameters than other competitors. Experimental results show that the proposed detector achieves competitive performance with other frameworks on MS COCO, PASCAL VOC 2007, and PASCAL VOC 2012. The PdNet offers the main advantage that it can run in real-time on a CPU by 11.9 FPS. In future work, the attention module can be applied in the detector to improve the accuracy by enhancing the specific human body feature without adding excessive computing power.

Acknowledgment

This results was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE)(2021RIS-003)

References

[1] X. Zhang, S. Cao, and C. Chen, "Scale-aware hierarchical detection network for pedestrian detection," *IEEE Access*, vol. 8, pp. 94 429–94 439, 2020.

[2] C. Vishnu, R. Datla, D. Roy, S. Babu, and C. K. Mohan, "Human fall detection in surveillance videos using fall motion vector modeling," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 17 162–17 170, 2021.

[3] A. Shahbaz and K.-H. Jo, "Deep atrous spatial features-based supervised foreground detection algorithm for industrial surveillance systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4818–4826, 2021.

[4] M. D. Putro, L. Kurnianggoro, and K.-H. Jo, "High performance and efficient real-time face detector on central processing unit based on convolutional neural network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4449–4457, 2021.

[5] S. Zhang, C. Bauckhage, and A. B. Cremers, "Efficient pedestrian detection via rectangular features based on a statistical shape model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 763–775, 2015.

[6] D. Chen, S. Xia, and Y. Zhou, "Pedestrian detection via contour fragments," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 4054–4060.

[7] Y. Xu, X. Cao, and H. Qiao, "A low-cost pedestrian detection system with a single optical camera," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 2, 2006, pp. 8759–8763.

[8] Y. Hsu and Y. Lin, "Adaptive fusion of multi-scale yolo for pedestrian detection," *IEEE Access*, vol. 9, pp. 110 063–110 073, 2021.

[9] W.-Y. Hsu and W.-Y. Lin, "Ratio-and-scale-aware yolo for pedestrian detection," *IEEE Transactions on Image Processing*, vol. 30, pp. 934–947, 2021.

[10] Q. Bai, J. Xin, H. Ye, Q. Wang, P. Shi, and S. Liu, "An efficient pedestrian detection network on mobile gpu with millisecond scale," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 3195–3199.

[11] C. B. Murthy and M. F. Hashmi, "Real time pedestrian detection using robust enhanced yolov3+," in *2020 21st International Arab Conference on Information Technology (ACIT)*, 2020, pp. 1–5.

[12] Q. Bai, J. Xin, M. Yan, Y. Wang, E. Li, and S. Zhao, "An optimized mask-guided mobile pedestrian detection network with millisecond scale," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 4975–4980.

[13] X. Shao, J. Wei, D. Guo, R. Zheng, X. Nie, G. Wang, and Y. Zhao, "Pedestrian detection algorithm based on improved faster rcnn," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5, 2021, pp. 1368–1372.

[14] X. Li, X. Luo, H. Hao, F. Chen, and M. Li, "Pedestrian detection method based on multi-scale fusion inception-ssd model," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, 2020, pp. 1549–1553.

[15] H. H. Nguyen, T. N. Ta, N. C. Nguyen, V. T. Bui, H. M. Pham, and D. M. Nguyen, "Yolo based real-time human detection for smart video surveillance at the edge," in *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, 2021, pp. 439–444.

[16] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving cnn feature extraction framework for mobile sensing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1441–1455, 2021.

[17] M. D. Putro, D.-L. Nguyen, and K.-H. Jo, "A dual attention module for real-time facial expression recognition," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 411–416.

[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[19] S. Li, Y. Li, Y. Li, M. Li, and X. Xu, "Yolo-firi: Improved yolov5 for infrared image object detection," *IEEE Access*, vol. 9, pp. 141 861–141 875, 2021.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[21] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, p. 303–338, 2010.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] J. B. Glenn Jocher, Alex Stoken, "ultralytics/yolov5: v3.0," Aug 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3983579.

[24] H. Zhai, J. Cheng, and M. Wang, "Rethink the iou-based loss functions for bounding box regression," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, 2020, pp. 1522–1528.