Advances in Technology Innovation, vol. 9, no. 1, 2024, pp. 42-49

Efficient Object Detection and Intelligent Information Display Using YOLOv4-Tiny

Ying-Tung Hsiao, Jia-Shing Sheu*, Hsu Ma

Department of Computer Science, National Taipei University of Education, Taipei, Taiwan, ROC

Received 04 August 2023; received in revised form 14 November 2023; accepted 15 November 2023

DOI: https://doi.org/10.46604/aiti.2023.12682

Abstract

This study aims to develop an innovative image recognition and information display approach based on you only look once version 4 (YOLOv4)-tiny framework. The lightweight YOLOv4-tiny model is modified by replacing convolutional modules with Fire modules to further reduce its parameters. Performance reductions are offset by including spatial pyramid pooling, and they also improve the model's detection ability for objects of various sizes. The pattern analysis, statistical modeling, and computational learning visual object classes (PASCAL VOC) 2012 dataset are used, the proposed modified YOLOv4-tiny architecture achieves a higher mean average precision (mAP) that is 1.59% higher than its unmodified counterpart. This study addresses the need for efficient object detection and recognition on resource-constrained devices by leveraging YOLOv4-tiny, Fire modules, and SPP to achieve accurate image recognition at a low computational cost.

Keywords: YOLOv4-tiny, deep learning, object detection, image recognition, information display

1. Introduction

Target detection methods that leverage deep learning have greatly advanced with the introduction of models such as recursive convolutional neural networks (R-CNNs) [1-2], Fast R-CNN [3], Faster R-CNN [4], Mask R-CNN [5], single shot detector (SSD) [6], and you only look once (YOLO) [7]. Mask R-CNN proposes a region of interest (ROI) align technology to address the problem that the bounding box positioning of Faster R-CNN's RoI pooling is not accurate enough. The ROI align uses bilinear interpolation to replace the original ROI pooling's use of integers to record coordinates and instead uses a floating point to record coordinates. SSD uses a multiBox detector based on a convolutional neural network and sets a series of anchor boxes (anchor boxes) on each layer of feature maps. Each anchor box corresponds to an object of different sizes and aspect ratios. The SSD detects and locates objects by predicting whether each anchor box contains an object and the category and location information of the object.

During training, SSD uses a cross-entropy loss function to minimize the gap between predicted values and true values. These methods have enabled accurate and efficient object detection in domains such as medical image analysis, face recognition, and self-driving cars [8]. However, applying computationally demanding deep learning models in resource-constrained edge devices is challenging [9-10]. In this study, the efficient and lightweight object detection architecture you only look at once version 4 (YOLOv4)-tiny [11] was modified to achieve a superior trade-off between accuracy and computational cost. Specifically, convolution-batch normalization-leaky ReLU (CBL) modules in YOLOv4-tiny were replaced with Fire modules to reduce the number of parameters without compromising performance [12], enabling its

* Corresponding author. E-mail address: jiashing@tea.ntue.edu.tw

English language proofreader: Chih-Wen Teng

deployment on low-power devices. Spatial pyramid pooling (SPP) was then to enhance the feature extraction capabilities of the network at various scales and capture fine-grained details in objects of varying sizes [13]. The proposed method was used to effectively recognize objects within images and could then display information on these objects.

The motivation behind this study is to assist individuals in capturing moments using their convenient photography devices or accurately retrieving desired product information from online photos. However, achieving high precision in image recognition and fast detection often relies on extensive computation and high-performance hardware. Therefore, an approach that maintains high accuracy while minimizing computational costs is investigated.

The remainder of this article is organized as follows: Section 2 presents a literature review related to the proposed method in the paper, Section 3 describes the system architecture of the modified YOLOv4-tiny architecture, which was adjusted to reduce the complexity of the CBL module, Section 4 outlines the experiments and presents the results, and Section 5 concludes the paper.

2. Literature Review and Methodology

The YOLO series is a well-established architecture and was selected as the basis for the proposed method [14-15]. YOLO is widely recognized for its real-time object detection capabilities and high efficiency. For YOLOv3, a novel backbone network, Darknet-53, was developed [16]. As the name suggests, Darknet-53 has 53 convolutional layers and uses the ResNet structure to solve the vanishing gradient problem. The architecture also uses feature pyramid networks (FPNs) in its neck; the different sizes of these FPNs improved its prediction accuracy for small targets. YOLOv4 [17] constitutes a leap in image recognition technology. Its backbone network, denoted CSPDarknet-53, is an integration of Darknet-53 and CSPNet. The FPN neck of YOLOv3 was replaced with SPP and a path aggregation network in YOLOv4. These changes both decreased its parameter count and improved detection accuracy, greatly increasing its average precision (AP) and frames per second (FPS) [11].

In this study, a version of YOLOv4, YOLOv4-tiny, was adopted to enable detection on computationally constrained hardware. Fig. 1 presents an overview of the YOLOv4-tiny architecture, which comprises a CSPDarknet53-tiny module, five convolution, batch normalization, and CBL modules, an upsampling layer, and two convolutional layers. Unlike YOLOv4, YOLOv4-tiny produces two feature outputs instead of three.

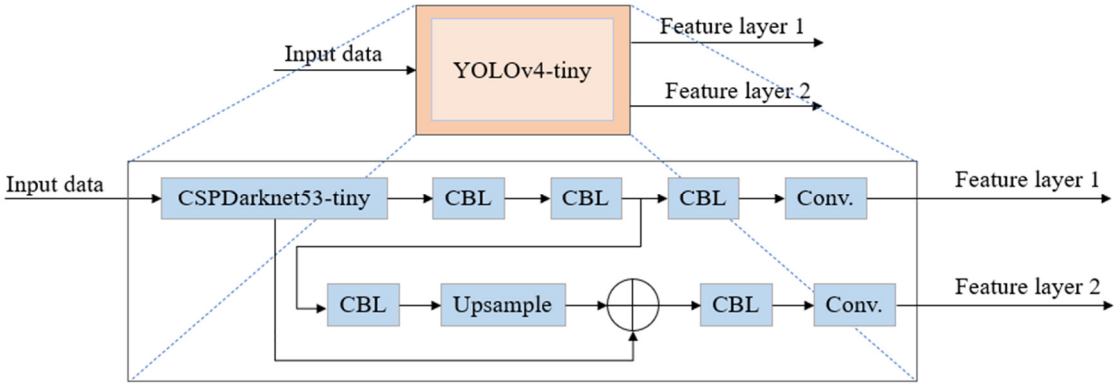

Fig. 1 YOLOv4-tiny architecture

The backbone of the YOLOv4-tiny architecture is the CSPDarknet53-tiny module shown in Fig. 2. This module comprises two CBL modules, three cross stage partial (CSP) modules, and three max pooling modules. In contrast to YOLOv4, which employs the Mish activation function, the CBL module in CSPDarknet53-tiny uses leaky ReLU for improved speed and performance. To enhance the efficiency of the CSP module, the original CSPnet [18] from YOLOv4 was modified by splitting the residual block into two parts, reducing intermediate processing; the outputs are merged at the end of the module shown in Fig. 3. This approach reduces the computational cost of the network architecture while improving accuracy.

Specifically, the input of the CSP module is a feature map, which is dimensionally reduced through a convolutional layer to generate two feature maps. After passing through some convolutional layers, batch normalization layers, activation function layers, etc., the two feature maps are concatenated and then passed through some convolutional layers to obtain the output feature map. Therefore, the CSP part can be seen as a multi-layer convolutional network that can learn high-level information from the input feature map, while the residual connection can retain low-level information from the input feature map, thus effectively utilizing the information in the input feature map.

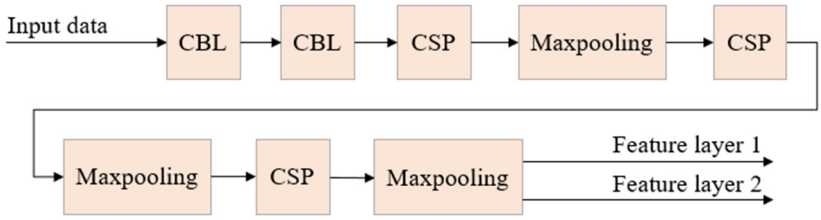

Fig. 2 CSPDarknet53-tiny module

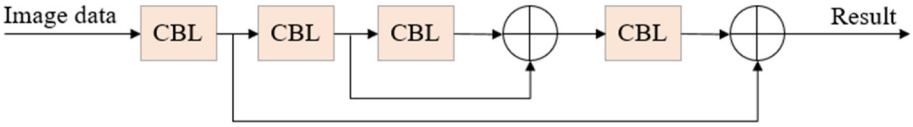

Fig. 3 CSP module

SPP was applied in this study to enhance the model's ability to detect objects at various scales [19]. Specifically, SPP facilitates feature extraction at different scales when the model is trained on images of various sizes. The proposed SPP module differs from previous approaches [19] in that it connects three feature maps of the same size to the input feature map, resulting in a final feature map four times the size of the original input as shown in Fig. 4. SPP mainly addresses two issues related to fully connected layers.

Firstly, fully connected layers require fixed-size inputs, which means that when using convolutional neural networks for object recognition, the size of the input image must be fixed, limiting the model's application range. Secondly, fully connected layers require a large amount of memory, which means that when the size of the input image increases, more memory is consumed, making it impossible to train the model or run it on devices with limited memory.

SPP solves these problems by pooling different-sized regions to obtain a fixed-size feature vector, which can be used as the input of the fully connected layer, avoiding the aforementioned problems. In addition, SPP can increase the model's ability to extract features of different scales. This is because, through pyramid pooling, SPP can extract features from regions of different sizes, thereby improving the model's generalization ability.

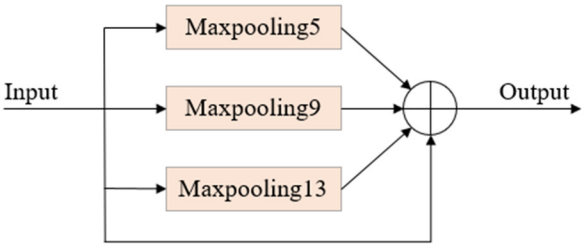

Fig. 4 SPP module

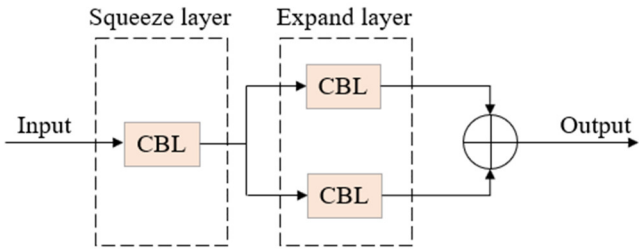

Fig. 5 Fire module

The fire module [12], comprising a squeezing part and two expansion parts, is crucial in reducing the model's parameter count. The squeeze part is a convolutional layer with a kernel size of 1, and the expansion part comprises two convolutional layers with kernel sizes of 1 and 3 shown in Fig. 5.

The following formulas represent the parameter calculation for a general convolutional layer (R_{conv}) and the fire module (R_{fire}), respectively.

$$R_{conv} = \left(C_{input} \times K^2 + 1\right) \times C_{output} \tag{1}$$

$$R_{fire} = \left(C_{input} \times K_{S1}^2 + 1\right) \times S1 + \left(S1 \times K_{E1}^2 + 1\right) \times E1 + \left(S1 \times K_{E3}^2 + 1\right) \times E3 \tag{2}$$

In Eq. (1), C_{input} is the number of channels of the current convolution input, K is the size of its kernel and C_{output} is the number of channels of its output. In Eq. (2), C_{input} is the input channel number of the fire module, K_{S1} is the kernel size of the squeeze layer, $S1$ is the output channel number of the squeeze layer, and the following parts are respectively the two convolution modules of the expand layer. K_{E1} is the kernel size of the convolution module, and $E1$ is its output channel number. Similarly, K_{E3} and $E3$ are both the kernel size and output channel number.

The customized YOLOv4-tiny architecture comprised the CSPDarknet53-tiny module, CBL modules, upsampling layer, and convolutional layers. The CBL module is mainly composed of a convolution layer, batch normalization, and an excitation function. The operation of the convolution layer mainly has the parameters, including convolution kernel size, stride, and padding, to determine the operation of the convolution and the control feature map.

In the convolution operation, by setting these parameters, the size, shape, depth, and other attributes of the output feature map can be controlled, so that the convolution layer can perform feature extraction and feature mapping of different sizes and directions on the input image. The modified SPP module and fire module were also included to improve feature extraction and reduce the model's parameter count. To evaluate the effectiveness of these modifications, the proposed model and vanilla YOLOv4-tiny were trained on the PASCAL VOC 2012 dataset and compared against each other.

3. System Structure

The YOLOv4-tiny architecture was modified to reduce the complexity of the CBL module. However, reducing parameters in the critical backbone of CSPDarknet53-tiny may reduce the accuracy of the model. According to Fang et al. [20], the CBL module following CSPDarknet53-tiny was replaced with an appropriate number of Fire modules. Eqs. (1)-(2) can be used to calculate the parameter counts of the CBL module and the modified Fire modules; the results for various modifications are presented in Table 1. Table 1 reveals that the Fire module reduced the number of parameters more for convolutional layers with more input channels. Consequently, two CBL modules were replaced with three Fire modules, and the CBL modules preceding the final output were replaced with two Fire modules, reducing the parameter count by a factor of 5.8. This reduction in parameters reduces accuracy; however, additional modules could be introduced to improve the model's multiscale feature extraction capabilities. Therefore, in reference to Prasetyo et al. [21], an SPP module was included after the Fire3 module to improve training stability for multisize training images.

Table 1 Parameters for various CBL and Fire module configurations and input/output channels

Input/output	CBL	Fire module
512/512	2359808	197184
512/256	131328	57632
256/256	-	49440
256/512	1180160	180800
256/256	-	197184
384/256	884992	53536
256/256	-	49440
Total	4556288	785216

The integrated definition for function modeling (IDEF) methodology is widely used in software development; IDEF0 was designed explicitly for functional modeling [22]. IDEF0 uses visual graphics and structured methods to describe the entire system, which can concisely and quickly understand the purpose and process of the system, effectively avoiding the use of excessive text descriptions and complicating the narrative. Fig. 6 presents an IDEF0 diagram that visually represents the key components of the proposed system architecture. In the image processing (A1) module, the images in the data set need to be pre-processed, including resizing and converting the image to grayscale for subsequent feature extraction and model training. Fig. 7 presents a diagram of the image processing and result presentation component, and Fig. 8 displays the modified YOLOv4-tiny architecture.

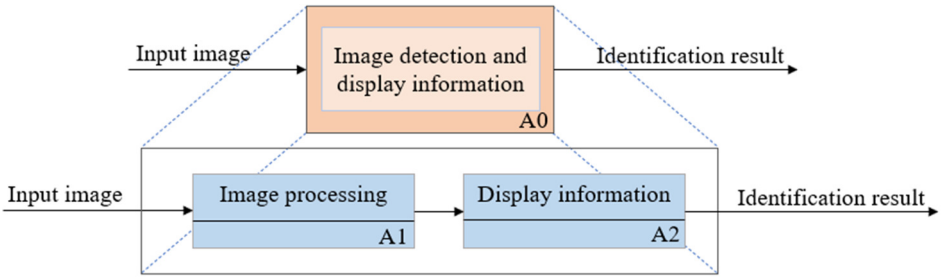

Fig. 6 IDEF0 for the image detection and information display system

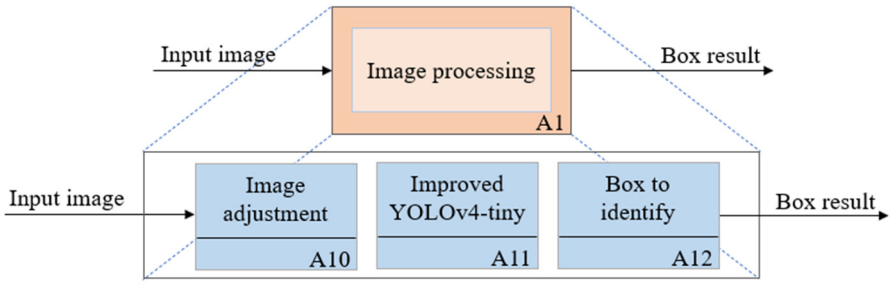

Fig. 7 IDEF0 for image processing

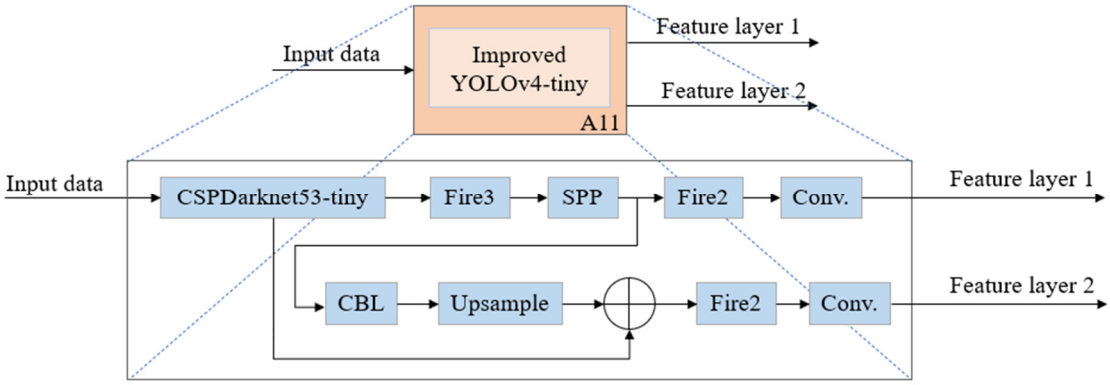

Fig. 8 IDEF0 for the improved YOLOv4-tiny network architecture

In summary, some CBL modules in YOLOv4-tiny were replaced with Fire modules to reduce their size and computational complexity; SPP was then incorporated to improve accuracy when training on images of different sizes. The IDEF0 diagram visually representing the image processing and information presentation components comprising the system architecture was also presented.

4. Experimental Results

This section presents the experiments' results to evaluate the proposed approach. The experiments were performed using an RTX 3060 Ti GPU with CUDA 11.3, significantly accelerating the training process. Detailed system configurations used in the experiments are provided in Table 2.

Table 2 Experiment types of equipment and system configurations

Components	Specification
Operating system	Windows 10
GPU	Nvidia GeForce RTX 3060 Ti
CPU	AMD Ryzen 9 5900X 12-Core
Memory	32GB
Cuda	11.3

The metrics for evaluating image recognition performance were precision, recall, and F1-score; these can be calculated from the confusion matrix. The confusion matrix is a representation of the correspondence between the actual and predicted labels of the input data as shown in Table 3.

Table 3 Confusion matrix

	True	False
True	True positive (*TP*)	False negative (*FN*)
False	False positive (*FP*)	True negative (*TN*)

The other metrics can be calculated from the confusion matrix,

$$\text{Precission} = \frac{TP}{TP+FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4}$$

$$\text{F1-score} = \frac{2\times \text{precission} \times \text{recall}}{\text{precission} + \text{recall}} \tag{5}$$

Eqs. (3)-(5) were applied to evaluate the performance of the modified and unmodified models. The modified model was called YOLOv4-tinyX1. The models were trained on the PASCAL VOC 2012 dataset, which has 20 object categories. YOLOv4-tinyX1 achieved higher precision than YOLOv4-tiny in each category presented in Table 4 but generally low recall presented in Table 5.

Table 4 Precision of YOLOv4-tiny and YOLOv4-tinyX1

	YOLOv4-tinyX1	YOLOv4-tiny
Aeroplane	95.77%	90.48%
Bicycle	74.42%	71.74%
Bird	82.50%	72.86%
Boat	78.38%	77.42%
Bottle	63.64%	72.73%
Bus	90.48%	81.63%
Car	81.36%	81.30%
Cat	82.18%	79.63%
Chair	64.29%	60.00%
Cow	73.08%	65.62%
Dining table	94.12%	73.91%
Dog	77.24%	71.97%
Horse	70.27%	68.57%
Motorbike	76.32%	75.68%
Person	82.33%	81.43%
Pottedplant	67.86%	64.29%
Sheep	78.43%	78.00%
Sofa	78.95%	75.00%
Train	91.11%	86.96%

Table 5 Recall of YOLOv4-tiny and YOLOv4-tinyX1

	YOLOv4-tinyX1	YOLOv4-tiny
Aeroplane	66.02%	73.79%
Bicycle	43.24%	44.59%
Bird	43.14%	33.33%
Boat	30.85%	25.53%
Bottle	11.97%	13.68%
Bus	70.37%	74.07%
Car	45.28%	47.17%
Cat	61.03%	63.24%
Chair	23.38%	22.08%
Cow	38.78%	42.86%
Dining table	21.62%	22.97%
Dog	62.91%	62.91%
Horse	35.14%	32.43%
Motorbike	50.88%	49.12%
Person	58.30%	60.08%
Pottedplant	20.43%	19.35%
Sheep	57.97%	56.52%
Sofa	29.41%	23.53%
Train	60.29%	58.82%

The YOLOv4-tinyX1 also outperforms YOLOv4-tiny in terms of F1-score in several categories presented in Table 6. The mean AP (mAP) of the models was also compared with an intersection over the union threshold of 0.5. Table 7 reveals that the training mAP of the YOLOv4-tinyX1 (57.03%) was superior to that of YOLOv4-tiny (52.93%); its testing mAP was also approximately 1.59% higher. The proposed model also achieved a lower giga floating point operations (GFLOPs) score, parameter count, and total size than YOLOv4-tiny, indicating that it was more efficient in terms of every metric.

Table 6 F1-Score of YOLOv4-tiny and YOLOv4-tinyX1

	YOLOv4-tinyX1	YOLOv4-tiny
Aeroplane	78%	81%
Bicycle	55%	55%
Bird	57%	46%
Boat	44%	38%
Bottle	20%	23%
Bus	79%	78%
Car	58%	60%
Cat	70%	70%
Chair	34%	32%
Cow	51%	52%
Dining table	35%	35%
Dog	69%	67%
Horse	47%	44%
Motorbike	61%	60%
Person	68%	69%
Potted plant	31%	30%
Sheep	67%	66%
Sofa	43%	36%
Train	73%	70%

Table 7 Overall results of YOLOv4-tiny and YOLOv4-tinyX1

	YOLOv4-tinyX1	YOLOv4-tiny
Train mAP	57.03%	52.93%
Test mAP	57.73%	56.14%
GFLOPs (G)	4.770	6.836
Total params (M)	2.107	5.876
Params size (MB)	8.04	22.42

These experimental results validate the effectiveness of the proposed modified architecture; it not only achieved higher precision, recall, and F1 scores in many or all categories but also higher mAP. It also had lower overall computational requirements and fewer parameters in every performance metric. Hence, despite its simpler architecture, the model was competitive with YOLOv4-tiny.

In summary, the experimental results substantiate the efficacy of the proposed modifications to YOLOv4-tiny. The achieved precision, recall, and mAP improvements highlight the proposed model's enhanced object detection capabilities. The reduction in computational costs, parameter size, and GFLOPs demonstrate the efficiency of the modified architecture.

5. Conclusions

A modified YOLOv4-tiny architecture was developed by incorporating Fire and SPP modules to replace some CBL modules, greatly reducing the parameter count while maintaining competitive performance. Including the SPP module enhanced feature extraction across different scales. In experiments, the modified architecture achieved training and testing mAPs that were 4.1% and 1.59% higher, respectively, than the unmodified YOLOv4-tiny architecture. These findings indicate that the proposed approach can achieve a superior trade-off between accuracy and computational cost relative to the original network; hence, it is a compelling solution for efficient object detection on resource-constrained devices, such as autonomous driving, surveillance systems, and other mobile applications with edge devices.

Future studies could further optimize the network by integrating additional lightweight modules or investigating alternative techniques to enhance computational efficiency while preserving accuracy. Additionally, the performance of the modified model could be evaluated on diverse datasets, and its scalability for larger applications could be assessed.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] M. S. B. Hossain, J. Dranetz, H. Choi, and Z. Guo, "DeepBBWAE-Net: A CNN-RNN Based Deep SuperLearner for Estimating Lower Extremity Sagittal Plane Joint Kinematics Using Shoe-Mounted IMU Sensors in Daily Living," IEEE Journal of Biomedical and Health Informatics, vol. 26, no. 8, pp. 3906-3917, August 2022.

[2] S. S. Islam, E. K. Dey, M. N. A. Tawhid, and B. M. M. Hossain, "A CNN Based Approach for Garments Texture Design Classification," Advances in Technology Innovation, vol. 2, no. 4, pp. 119-125, October 2017.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, June 2017.

[4] W. Wu, Y. Yin, X. Wang, and D. Xu, "Face Detection with Different Scales Based on Faster R-CNN," IEEE Transactions on Cybernetics, vol. 49, no. 11, pp. 4017-4028, November 2019.

[5] X. Bi, J. Hu, B. Xiao, W. Li, and X. Gao, "IEMask R-CNN: Information-Enhanced Mask R-CNN," IEEE Transactions on Big Data, vol. 9, no. 2, pp. 688-700, April 2023.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, et al., "SSD: Single Shot MultiBox Detector," Computer Vision – ECCV 2016: 14th European Conference on Computer Vision, pp. 21-37, October 2016.

[7] K. H. Tseng, M. Y. Chung, W. H. Jhou, W. J. Chang, and C. H. Xie, "Development of Non-Contact Real-Time Monitoring System for Animal Body Temperature," Proceedings of Engineering and Technology Innovation, vol. 21, pp. 27-33, April 2022.

[8] D. Zhu, G. Xu, J. Zhou, E. Di, and M. Li, "Object Detection in Complex Road Scenarios: Improved YOLOv4-Tiny Algorithm," 2nd Information Communication Technologies Conference, pp. 75-80, May 2021.

[9] M. B. Ullah, "CPU Based YOLO: A Real Time Object Detection Algorithm," IEEE Region 10 Symposium (TENSYMP), pp. 552-555, June 2020.

[10] S. Chen and W. Lin, "Embedded System Real-Time Vehicle Detection Based on Improved YOLO Network," 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference, pp. 1400-1403, October 2019.

[11] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13029-13038, June 2021.

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5 MB Model Size," https://doi.org/10.48550/arXiv.1602.07360, November 04, 2016.

[13] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection," Information Sciences, vol. 522, pp. 241-258, June 2020.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, June 2016.

[15] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," IEEE Conference on Computer Vision and Pattern Recognition, pp. 6517-6525, July 2017.

[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", https://doi.org/10.48550/arXiv.1804.02767, April 08, 2018.

[17] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection," https://doi.org/10.48550/arXiv.2004.10934, April 23, 2020.

[18] C. Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 1571-1580, June 2020.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904-1916, September 2015.

[20] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," IEEE Access, vol. 8, pp. 1935-1944, 2020.

[21] E. Prasetyo, N. Suciati, and C. Fatichah, "YOLOv4-Tiny and Spatial Pyramid Pooling for Detecting Head and Tail of Fish," International Conference on Artificial Intelligence and Computer Science Technology, pp. 157-161, June 2021.

[22] J. S. Sheu and C. Y. Han, "Combining Cloud Computing and Artificial Intelligence Scene Recognition in Real-Time Environment Image Planning Walkable Area," Advances in Technology Innovation, vol. 5 no. 1, pp. 10-17, January 2020.