Embedded Real-Time Pedestrian Detection System Using YOLO Optimized by LNN

1st Yuanzhe Jin
Northwestern University
Evanston, USA
yzjin@u.northwestern.edu

2nd Yixun Wen
Georgia Institute of Technology
Atlanta, USA
ywen75@gatech.edu

3rd Jingting Liang
Harvard University
Cambridge, USA
jingting_liang@g.harvard.edu

***Abstract*—The main challenge for real-time pedestrian detection in an embedded system is to keep a balance between accuracy and computation cost. To solve this, the main idea is to use Lightweight Neural Network (LNN), a special design Neural Networks which can provide acceptable accuracy with faster speed.**

In this paper, we modify the YOLO written in Darknet with MobileNet, one of the popular LNN in the embedded system. And we realize deploying real-time pedestrian detection using the modified YOLO on the platform Jetson TX2.

***Index Terms*—Pedestrian Detection; Embedded Systems; Lightweight Neural Networks; YOLO Object Detection; MobileNet**

I. Introduction

Most of the current pedestrian detection devices perform calculations by sending image data to a cloud server, and sending them back to the device after the cloud computing is complete. This process transfers the computing task to a computing device located in the cloud and uses its calculations to avoid detecting the lack of computing power of the device. But this requires that the running equipment and processing cloud are in a low-latency network. 5G technology may be a feasible idea to solve this problem in the future [1]. At present, if only embedded devices are used for arithmetic processing, it is still difficult to achieve accurate and real-time pedestrian detection.

Reliable pedestrian detection technology has been a need to ensure autonomous vehicles that can have a real-time understanding of the surrounding pedestrian conditions [2]. Due to the difference in computing power between embedded devices and large servers, it cannot handle a large number of convolution calculations. So there needs to be a simplified version of the neural network to meet the needs of the embedded platform.

In view of this problem, we modify the existing pedestrian detection neural network and transplant it to the embedded platform to implement the pedestrian detection function on the embedded platform.

II. Related Work

Researchers have been studying pedestrian detection for a long time. YOLO is one of the most advanced real-time object detection systems currently widely used, especially in autonomous driving [3]. YOLO has experienced development from YOLOv1 to YOLOv3. Redmon and Farhadi first proposed the advantages of YOLO in processing images compared with the current multiple algorithms [4]. This article introduces YOLO 9000 and YOLOv2, both of which can be used to detect target objects. The difference is that YOLO9000 can detect more than 9,000 different objects, while YOLOv2 continues to develop on the basis of YOLOv1. The recognition accuracy of objects is higher, and the same type of recognition objects is inferior to YOLO9000. Redmon and Farhadi in their other paper [5], proposed a further upgraded version of YOLOv2 YOLOv3. Compared with YOLOv2, YOLOv3 has a more complex structure and higher recognition accuracy.

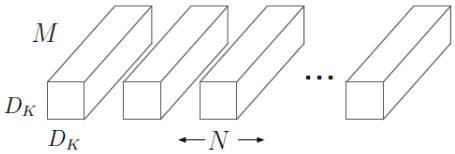
For complex neural network structures like YOLOv3, Liu, Anguelov, and Erhan proposed a method that uses a single method to detect objects in an image called Single Shot Multi-Box Detecton (SSD) to simplify the neural network structure [6]. SSD is more convenient during training and can simplify the network. When using the SSD network for prediction, the network compares the generated predicted image frame with the actual image frame, and adjusts to obtain better results. SSD also has a certain object prediction function, which can deal with problems of different sizes [7]. This makes SSD training very valuable and practical. It can be integrated into the system for common training to complete.

Similar to the idea of Liu, Ren and He proposed a new neural network called Fast R-CNN [8]. Fast RCNN reduces the running time of the detection network. Using the newly proposed Regional Proposal (RPN) method by the authors and others can share convolution features during training. RPN is trained end-to-end and can produce high-quality results for fast R-CNN detection.

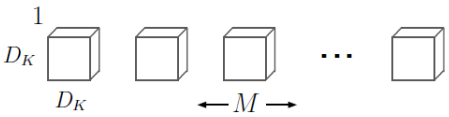
Based on previous research by researchers, if the researchers want to perform YOLO operations on embedded devices, they need to simplify the YOLO network appropriately. At present, there are two main methods for the simplification of the YOLO neural network. One is the use of TensorRT developed by Nvidia and optimized for Nvidia GPUs. The TensorRT instruction manual details the use of TensorRT [9]. The entire process can be summarized into three steps, namely model analysis, engine optimization, and final execution. Based on the instruction manual, the researchers further proposed an optimization scheme for deep learning using TesnorRT [10].

For the simplification of YOLO, TensorRT can be used for reference, so that the calculation of YOLO on the GPU of the embedded system can be optimized.However, TensorRT only supports ONNX, Caffe and Uff (Universal Framework Format) currently, which cannot be applied to most of the embedded systems.

The other way to simplify the YOLO network is using LNN. LNN is a large group of neural network using methods to reduce calculation in the processing. There are many LNN like LiteflowNet [11], MobileNet [12], SqueezeNet [13], Xception [14], ShuffleNet [15] and etc. ShuffleNet also has its updated version called ShuffleNetV2 [16]. The core idea of the MobileNet is to use a computation called separable convolutional groups [12]. As can be seen from the network name, this network is mainly used with mobile devices [17]. The method of calculating convolution in MobileNet network uses the steps of "Separation-Calculation-Merging". By separating the original 3×3 convolution into three 1×1 convolutions, the repeated calculation of the convolution kernel is reduced. This reduces the amount of calculations and the number of parameters required [18].



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

Fig. 1. MobileNet Network Structure [12].

For MobileNet improvement, there are also some strategies [19]. Researchers change its network architecture and reduce its calculation amount to make it easy to deploy on the embedded platform. MobileNet, as one of the most popular LNN in the research, is taken as the main method in this paper. It is also wildly used in many scenarios like garbage classification [20], skin cancer detection, palmprint recognition [21] and etc.

In this paper, we just talk about using original MobileNet's method, and avoid using some improved MobileNet versions, which still needs some time to test their effectiveness and reliability.

III. Development Platform

The development platform used in this research is the Jetson TX2, which is produced by Nvidia. Jetson TX2 has strong terminal computing capabilities and can rely on its own hardware for calculations. The Jetson TX2 carries NVIDIA's Tegra processor, which helps it have powerful calculation ability on image processing and face recognition [22]. The comparison of Jetson TX1 and Jetson TX2 is shown in Fig.1. Compared with Jetson TX1, Jetson TX2 has greater improvement in GPU computing power, memory and computing speed. Besides, Jetson TX2 is often equipped with an Ubuntu system, which can easily install a variety of software-based on the Linux platform.

	Jetson TX2	Jetson TX1
GPU	NVIDIA Pascal™, 256 CUDA cores	NVIDIA Maxwell ™, 256 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2	Quad ARM® A57/2 MB L2
Video	4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)	4K x 2K 30 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (10-Bit Support)
Memory	8 GB 128 bit LPDDR4 59.7 GB/s	4 GB 64 bit LPDDR4 25.6 GB/s
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4	2x DSI, 1x eDP 1.4 / DP 1.2 / HDMI
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.1 (1.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2	Gen 2 | 1x4 + 1x1
Data Storage	32 GB eMMC, SDIO, SATA	16 GB eMMC, SDIO, SATA
Other	CAN, UART, SPI, I2C, I2S, GPIOs	UART, SPI, I2C, I2S, GPIOs
USB	USB 3.0 + USB 2.0	
Connectivity	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth	
Mechanical	50 mm x 87 mm (400-Pin Compatible Board-to-Board Connector)	

Fig. 2. Performance Comparison of Jetson TX2 and Jetson TX1

IV. Optimize Network Architecture

The difficulty of modifying YOLO network is because of its unique architecture, Darknet. Although it lacks the readability and portability of Python, Darknet has the characteristics of fast installation, support for CPU and GPU computing, and fast deployment. Unlike common neural network structures in Python, such as TensorFlow, Pytorch, or Caffe, Darknet is an open source neural network framework written in C and CUDA. It helps to reach a higher calculation speed [23] but it causes the problem for other researchers to improve it. The key to improve YOLO's speed is to apply lightweight networks in Darknet. The original YOLO network structure is shown in Fig.3.

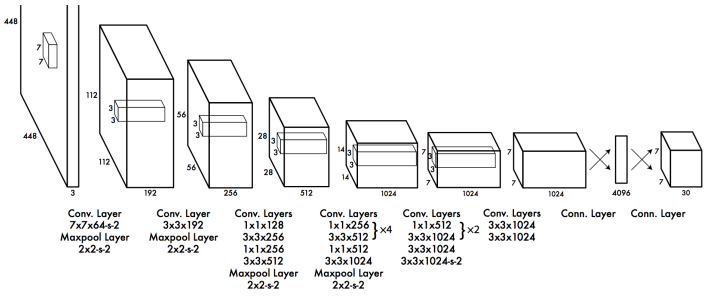


Fig. 3. YOLO Network Structure [4]

Besides the architecture shows in the Fig.3., an important concept in YOLO is Anchor. The number of Anchors is the same as the maximum number of object types (types) that can appear in each grid. In actual operation, the number of Anchors can be greater than the number of detected objects. The value of Anchor is generally placed at the end of the

network, and the specific value of Anchor can be automatically calculated by the running script. Anchor appeared to solve the situation where there are multiple target objects in the same grid [24]. If the division of a network is reduced, the situation where there are multiple target objects in the same grid will be less frequent. There are similarities in the simplified ideas of lightweight neural networks, all of which are to reduce the internal redundancy of the convolutional layer and improve the computing performance and network performance.

YOLO uses multi-layer convolutions for image detection, of which 3 × 3 convolutions occupy a major part of the amount of calculation. By analyzing light-weight networks such as MobileNet, it can be found that the core of the reduction in light-weight network calculations lies in deep separable convolutions. After using a lot of 3 × 3 convolutional layers, they are followed by 1 × 1 convolutional layers to integrate different channel information.

Considering the number of filters in the output layer, the network structure can be preserved as much as possible by reducing the convolution dimension of each layer. As the core idea of lightweight networks such as MobileNet, deep separable convolution structure (Depthwise Convolution). This method can achieve a deep convolution structure after the previous convolution connection on the premise of keeping the neural network channels separated spatial convolution.

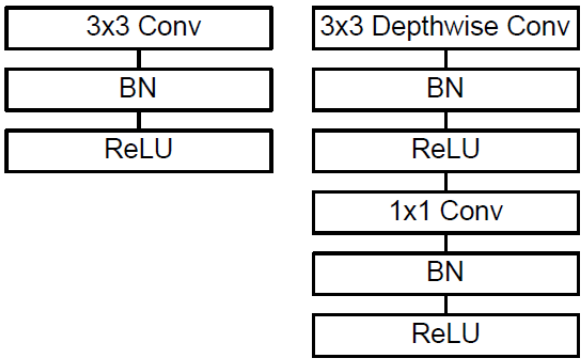


Fig. 4. The YOLO network structure is on the left, and the rewritten YOLO network structure is on the right [12].

We optimized the YOLO archtecture based on the thinking of MobileNet as it is shown in Fig.4. The idea of using Mobilenet is of great significance to improve the calculation speed of YOLO. In MobileNet, the idea is to use 3 × 3 convolution + BN (Batch Normalization) + ReLU (activation function) + 1 × 1 convolution + BN + ReLU. In contrast, the idea of YOLO thought convolution layer + BN + pooling layer + Leaky in Darknet. The YOLO network in the original Darknet was rewritten to a lightweight neural network using the idea of MobileNet by modifying, so as to increase the calculation speed within an acceptable range of accuracy loss. As can be seen from the figure, the rewritten new YOLO network on the right uses a 3 × 3 convolution network rewritten to a 1 × 1 convolution network, thereby achieving the purpose of reducing the originally required calculation amount. In this study, the activation function originally used in MobileNet was replaced with that used in YOLO. The choice of activation function will lead to different network effects.

The original YOLO network used the Leaky activation function, while the MobileNet-YOLO network used the ReLU activation function. Both are ”unsaturated activation functions”. The ReLU function was first proposed by Griffin Huntington. It sets all negative values in the matrix x to zero, leaving the rest unchanged. The calculation is performed after the convolution calculation, so it is calculated in the same order as the tan (h) function and the sigmoid function. Leaky ReLU (Leaky for short) activation function is developed on the basis of ReLU function. It gives all negative values a non-zero slope. The use effect is better on a small training set, but in actual situations, it is still the most used ReLU activation functions.

Using the idea from Fig.4., we combine the original YOLO with MobileNet and get a new LNN called YOLO-MobileNet. Its archtecture is shown in the Fig.5.

We modify the YOLO in the second 3 × 3 layer into a 3 × 3 depthwise layer and a 1 × 1 layer. It can reduce the calculation efficiently especially in the amount of parameters.

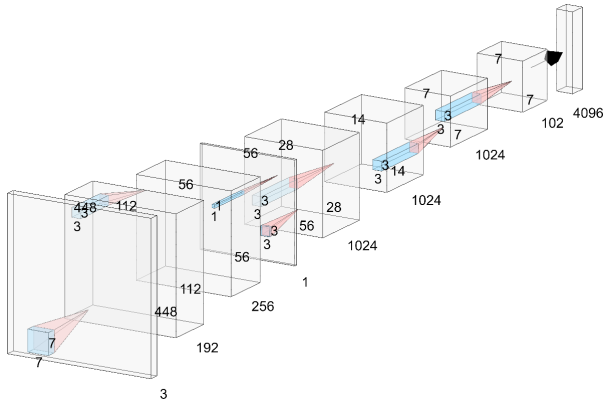


Fig. 5. YOLO-MobileNet Architecture

The amout of parameters calculation equation in a Neural Network can be presented as the following:

$$P_{sum} = \sum (T_{in} \cdot T_{out} \cdot C_k \cdot n) \tag{1}$$

In this equation, P_{sum} represents the sum of parameters of a certain Neural Network, T_{in} represents the number of input tunnels, T_{out} represents the number of output tunnels, C_k means the number of convolution kernels, and n means the depth of convolution layers.

Take an actual network example. Suppose a neural network consists of a 3 × 3 convolution layer, with 32 input channels and 64 output channels. Specifically, 64 of 3 × 3 convolution kernels will traverse each data in 32 channels, thereby generating 32 × 64 = 2048 feature maps. Feature map calculated by each input channel to obtain 64 output channels after filtering using a filter

For this example, we use deep separable convolutions and use 32 of 3 × 3 convolution kernels to traverse 32 channels of data to obtain 32 feature maps. Then use the 64 of 1 × 1 size convolution checks to traverse the previously generated feature maps. After this operation, we can get parameters of YOLO-MobileNet is equal to 2624, which is lower than the 18432 parameters required in a general network.

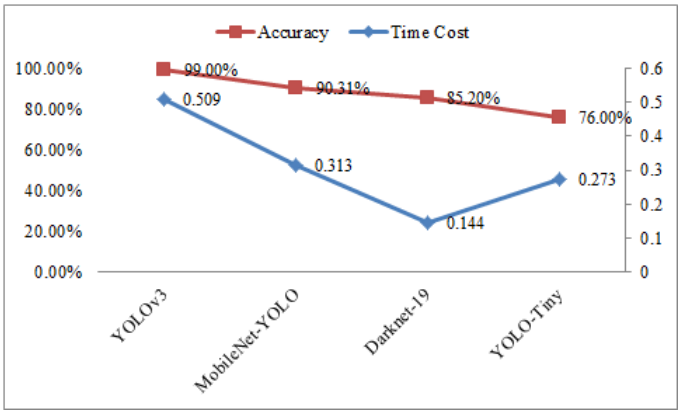


Fig. 6. The comparison of time and accuracy of different neural networks under the Darknet network (Time Unit: second).

In this research, the four network ideas are unified and rewritten to use Darknet as a model of network structure. Among them, YOLOv3, Darknet-19, and YOLO-Tiny are all under the original Darknet network, while MobileNet-YOLO is the result of rewriting by borrowing lightweight network ideas in this study.

As can be seen from Fig.6., in the case of processing a single picture, the running speed is Darknet-19, YOLO-Tiny, MobileNet-YOLO, and YOLOv3 in order from fast to slow. It can be seen that by using the idea of a lightweight network, the calculation amount can be reduced and the calculation speed can be increased. Under the same platform, the rewritten MobileNet-YOLO has a nearly 40% improvement in operating speed compared to the original YOLOv3.

Since this paper's research is mainly aimed at improving YOLOv3, it is mainly compared with YOLOv3 in the final accuracy comparison. The improved MobileNet-YOLO network has reduced its accuracy by approximately 20% and has increased its calculation speed by approximately 40%. It shows that the modification of neural networks using lightweight network ideas is effective. It can achieve a certain improvement in helping the algorithm better realize the transplantation of the embedded platform.

V. Conclusion

The significance of this research lies in deploying pedestrian detection on the embedded platform. The algorithm accelerates the calculation speed to about 40%, which can improve the original YOLOv3 speed from 6 frames per second to 9-10 frames per second. To further improve the results in real life, some techniques can be adopted through embedded devices. For example, modifying the resolution of the input image can greatly improve performance, but there is a problem of image distortion.

The Jetson TX2 currently used has Nvidia's GPU as the core of graphics processing and computing. On smaller platforms such as the Raspberry, camera and other non-GPU platforms, how to run neural networks on these devices with more computing power it is a future research direction. The realization of miniaturized pedestrian detection will be able to help many future production and lifestyle, which has broad application prospects.

References

[1] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From iot to 5g i-iot: The next generation iot-based intelligent algorithms and 5g technologies," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 114–120, OCTOBER 2018.

[2] M. Kilicarslan, J. Y. Zheng, and K. Raptis, "Pedestrain detection from motion," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 1857–1863.

[3] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–10.

[4] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] ——, "Yolov3: An incremental improvement," vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," 2016, to appear. [Online]. Available: http://arxiv.org/abs/1512.02325

[7] Y. Li, H. Huang, Q. Xie, L. Yao, and Q. Chen, "Research on a surface defect detection algorithm based on mobilenet-ssd," *Applied Sciences*, vol. 8, no. 9, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/9/1678

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf

[9] S. O. Settle, M. Bollavaram, P. D'Alberto, E. Delaye, O. Fernandez, N. Fraser, A. Ng, A. Sirasao, and M. Wu, "Quantizing convolutional neural networks for low-power high-throughput inference engines," *CoRR*, vol. abs/1805.07941, 2018. [Online]. Available: http://arxiv.org/abs/1805.07941

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[11] T.-W. Hui, X. Tang, and C. Change Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *CoRR*, vol. abs/1707.01083, 2017. [Online]. Available: http://arxiv.org/abs/1707.01083

[16] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *The European Conference on Computer Vision (ECCV)*, September 2018.
[17] H. Chen and C. Su, "An enhanced hybrid mobilenet," in *2018 9th International Conference on Awareness Science and Technology (iCAST)*, Sep. 2018, pp. 308–312.
[18] J. Su, J. Faraone, J. Liu, Y. Zhao, D. B. Thomas, P. H. W. Leong, and P. Y. K. Cheung, "Redundancy-reduced mobilenet acceleration on reconfigurable logic for imagenet classification," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, N. Voros, M. Huebner, G. Keramidas, D. Goehringer, C. Antonopoulos, and P. C. Diniz, Eds. Cham: Springer International Publishing, 2018, pp. 16–28.
[19] Z. Qin, Z. Zhang, X. Chen, C. Wang, and Y. Peng, "Fd-mobilenet: Improved mobilenet with a fast downsampling strategy," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 1363–1367.
[20] S. L. Rabano, M. K. Cabatuan, E. Sybingco, E. P. Dadios, and E. J. Calilung, "Common garbage classification using mobilenet," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, Nov 2018, pp. 1–4.
[21] S. S. Chaturvedi, K. Gupta, and P. S. Prasad, "Skin lesion analyser: An efficient seven-way multi-class skin cancer classification using mobilenet," *ArXiv*, vol. abs/1907.03220, 2019.
[22] E. Jose, G. M., M. T. P. Haridas, and M. H. Supriya, "Face recognition based surveillance system using facenet and mtcnn on jetson tx2," in *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*, March 2019, pp. 608–613.
[23] U. Upadhyay, F. Mehfuz, A. Mediratta, and A. Aijaz, "Analysis and architecture for the deployment of dynamic license plate recognition using yolo darknet," in *2019 International Conference on Power Electronics, Control and Automation (ICPECA)*, Nov 2019, pp. 1–6.
[24] B. Chen and X. Miao, "Distribution line pole detection and counting based on yolo using uav inspection line video," *Journal of Electrical Engineering Technology*, vol. 15, pp. 441–448, 2019.
[25] S. Paisitkriangkrai, C. Shen, and A. v. d. Hengel, "Pedestrian detection with spatially pooled features and structured ensemble learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1243–1257, June 2016.
[26] H. Ullah, N. Gopalakrishnan Nair, A. Moore, C. Nugent, P. Muschamp, and M. Cuevas, "5g communication: An overview of vehicle-to-everything, drones, and healthcare use-cases," *IEEE Access*, vol. 7, pp. 37 251–37 268, 2019.