This article has been accepted and published on J-STAGE in advance of copyediting. Content is final as presented.

IEICE Electronics Express, Vol.VV, No.NN, 1–5

LETTER

High-accuracy Low-latency Non-Maximum Suppression Processor for Traffic Object Detection

Chenbo Yuan[1,2,3], Peng Xu[1], and Gang Chen[1,2,3]

Abstract As autonomous driving technology advances, the requirements for object detection are becoming increasingly high. Non-maximum suppression (NMS) algorithm, as a key component in traffic object detection algorithms, is an independent post-processing process in the object detection framework. Due to the complexity of real-world road scenarios and high density of detected entities in urban traffic, the number of candidate bounding boxes generated by the neural network is large. Hence, low-precision processors may generate a significant number of redundant target bounding boxes. The excessive output of redundant target bounding boxes not only imposes a workload on subsequent processing but also has the potential to result in non-optimal decision-making. We propose a high-performance NMS processor that can quickly process a large number of candidate boxes without performing sorting of their scores. Also, it has low precision loss computing units and high parallel computing arrays. Combined with algorithm design, it effectively reduces the computational complexity and reduces the inference time of the end-to-end task of the NMS algorithm. Thus, our NMS processor's speed is comparable to SOTA architecture, and the average accuracy loss is only 0.4% .

key words: Traffic object detection, NMS algorithm, hardware accelerator.
Classification: Integrated Circuit

1. Introduction

Autonomous driving is gaining prominence in daily life, driving the research and development of multi-object detection algorithms with high accuracy and low latency[1, 2, 3, 4, 5, 6]. NMS is a crucial step in these algorithms that directly affects the output results. However, NMS is a greedy algorithm that is computationally intensive and has a complexity of $O(N^2)$, leading to increased processing time for a large number of detected targets.

Recent many FPGA-based and ASIC edge neural network acceleration chips[7, 8, 9, 10, 11, 12, 13, 14] such as UNPU[11], Eyeriss[12], and CASSANN-v2[13], have been proposed to target general neural network operations (i.e., convolution). However, when deploying object detection neural networks, these chips often offload the NMS algorithm to the on-chip embedded CPU, significantly increasing the end-to-end inference time of object detection neural networks at the edge.Therefore, it is vital to develop a customized circuit to reduce the computation time of the NMS algorithm at the edge.

In recent years, researchers have proposed various NMS acceleration architectures[15, 16, 17, 18, 19, 20] to tackle the slow speed of the NMS algorithm on general-purpose processors by simplifying the calculations. Some approaches[15, 16, 18, 19, 20] are suitable for simple applications like face detection and ship detection, where sacrificing precision for speed is acceptable. Some approaches[17] pursue excessive accuracy and ignore processing speed, which cannot be applied to applications with high real-time requirements. Therefore, they may not be well-suited for complex applications like autonomous driving. In such tasks, further processing is critical to avoid an increased number of redundant target bounding box outputs, because redundant target bounding boxes can impose a higher processing burden on subsequent processors and lead to irrational decisions made by subsequent processors. Therefore, to reliably handle such tasks at the edge, there is a need for alternative approaches that balance computational efficiency and accuracy.

In contrast to the previous NMS accelerator architecture, our primary focus lies on elevating the precision of the NMS processor, while simultaneously ensuring its processing speedIt is capable of effectively addressing post-processing requirements in intricate application scenarios, thereby mitigating the propagation of errors or redundant outputs to subsequent processing units. Furthermore, this approach aids in minimizing safety risks and alleviating the workload on the subsequent processing system. The key contributions of this work can be summarized as follows:

1) Through algorithm and hardware co-design, we have proposed an NMS processor that can efficiently process a large number of candidate boxes with high precision and speed.
2) To efficiently accelerate the NMS algorithm with low precision loss, a series of computation unit circuits and parallelization architecture are designed.

[1]Institute of Semiconductor, Chinese Academy of Sciences, Beijing, 10083,China
[2]University of Chinese Academy of Sciences, Beijing, 10083,China
[3]Semiconductor Neural Network Intelligent and Computing Technology Beijing Key Laboratory, Beijing 100083, China
a) chengang08@semi.ac.cn

DOI: 10.1587/elex.20.20230445
Received September 28, 2023
Accepted October 16, 2023
Publicized October 25, 2023

Copyright © 2023 The Institute of Electronics, Information and Communication Engineers

3) To improve overall efficiency, a set of macro operators are defined to optimize data flow and task scheduling, enhance data reuse, and increase cache locality.

2. Non-Maximum Suppression Analysis

NMS is a post-processing algorithm for object detection. It filters redundant bounding boxes, keeping only the highest confidence and non-overlapping detections. The steps include sorting boxes by confidence, selecting the highest confidence box, and removing overlapping boxes based on IoU threshold. The result is a set of independent detections. Although some of the current improved NMS algorithms[21, 22, 23, 24, 25, 26, 27, 28] add complex logic calculations to improve the reliability of filtering redundant bounding boxes, these algorithms are not hardware-friendly Before performing the IoU calculations, the candidate bounding boxes are first sorted by their confidence scores in the traditional NMS algorithm. This sequential approach is not suitable for efficient data flow in the overall system. The sorting step creates a significant amount of data movement overhead, which leads to an increase in power consumption and latency.

In the NMS algorithm, the significance of confidence score comparison between candidate bounding boxes only arises when their IoU exceeds a threshold. Hence, there is a rational basis to prioritize the computation of IoU calculation over the sorting operation. The transformed NMS algorithm is illustrated in Algorithm 1. Despite the increased computational workload resulting from prioritizing IoU calculations, this strategic choice facilitates more efficient data flow within the processor, leading to decreased data movement and improved processing speed. Moreover, by adhering to the transformed NMS algorithm, it becomes feasible to integrate IoU calculations and confidence score comparisons within the design of the computational unit, which can reduce area overhead.

Algorithm 1 Transformed NMS Algorithm

```
NMS(B, S, threshold, Table)
    N = Batch * sizeof(RAM)
    Table =Ones(N)
    for( k = 0; k <Batch ; k++ )
    for( n = k; n <Batch ; n++ )
    for( i = 0; i < sizeof(RAM) ; i++ )
    for( j = i + 1; j < sizeof(RAM); j++ )
        if IoU(bi, bj) > threshold and si > sj
            Table[n * sizeof(RAM)+j] = 0,
        else if IoU(bi, bj) > threshold and si < sj
            Table[k * sizeof(RAM)+i] = 0
    return Table
```

3. Overall Architecture

In order to reduce hardware resource overhead and improve data reuse, we adopt a bit table-based approach in the architecture design to record the status of corresponding candidate bounding boxes. As depicted in Fig.1, compared to the previous architecture design, we have made several improvements in the NMS processor. Firstly, to increase the processing capacity for candidate bounding boxes, we allocate sufficient space in external memory to store more candidate box status. By updating the on-chip box bit-table (BBT) register from external memory, we are able to reduce the on-chip storage resource. Secondly, in order to enhance processing accuracy, we create high-precision parallel computation units(PCUs). Lastly, to improve processing speed and data flow efficiency, we optimize data reuse and minimize data movement through the design of network-on-chip units (NOC) and the control unit.

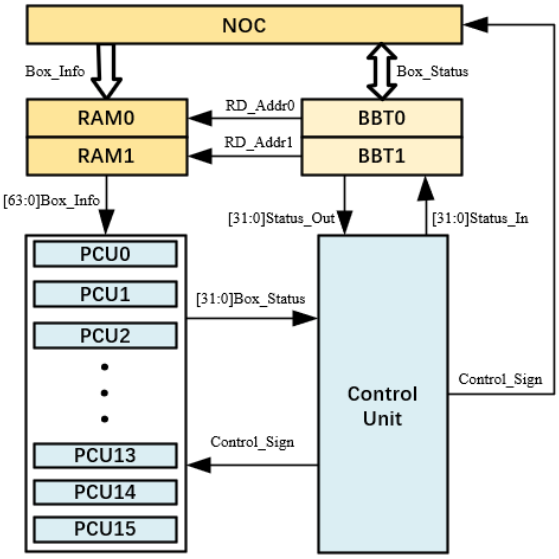


Fig. 1. The architecture of NMS processor

The BBT0 and BBT1 in Fig.1 respectively cache the status of candidate bounding boxes in the RAM0 and RAM1. The status of each candidate bounding box is represented by either "0" or "1". "0" means the candidate bounding box is to be discarded, and "1" means it should be retained. In addition, the RAM0 and RAM1 in Fig.1 are two partitions of a single RAM, storing candidate bounding box data for two different batches. Consequently, the BBT0 and BBT1 are two partitions of a single register.

The data flow of the NMS processor mainly consists of the following three processes:

Step 1: The NOC reads candidate bounding box data from two different batches and stores them in RAM0 and RAM1.

Step 2: 16 PCUs sequentially read the 16 candidate bounding boxes that have not been removed from RAM0.

Step 3: RAM1 simultaneously sends the remaining one of candidate bounding boxes to the 16 PCUs.

When the NMS processor is activated, these three main steps are executed sequentially. Once Step 3 completes traversing the data in RAM1, it goes back to executing Step 2 and Step 3. When all candidate bounding box data in RAM0

has been traversed, the processor returns to executing Step 1, 2, and 3 until all batches from external memory have been processed. Notably, the whole BBT register undergoes continuous updates by PCUs throughout the computational Step. Moreover, when the NMS processor returns to execute Step 1, the current BBT data needs to be uploaded to external memory, and the status data of the subsequent two batches needs to be downloaded for further processing.

Our NMS processor can be easily integrated into any system that necessitates NMS processing. For any system, when it asks to perform NMS processing, it only needs to write all candidate bounding boxes into the external memory and allocate a block of external memory space to store the status values of all candidate bounding boxes. The NMS processor does not perform any additions or deletions to the candidate bounding box data. It simply updates their status values to indicate whether the corresponding candidate bounding boxes should be retained or removed. Therefore, when the NMS processor completes its processing, the system can read the corresponding candidate bounding boxes based on their status values in the external memory for subsequent processing.

3.1 parallel computing array

The design of the computation array is crucial for achieving a balance between low precision loss and high-speed computation. The prior NMS processor architecture [15, 16] utilized an approximate IoU calculation methodology to design computation units, effectively reducing critical path delays but at the cost of diminished precision. In contrast, our proposed NMS processor's computation units, as depicted in Fig.2, do not use approximate IoU calculation. Instead, our approach involves targeted optimizations to minimize critical path delays while ensuring both high precision and processing speed for the processor.

$$IoU(a,b) = \frac{A(a \cap b)}{A(a) + A(b) - A(a \cap b))} \geq threshold. \tag{1}$$

$$\Rightarrow \frac{A(a \cap b)}{A(a) + A(b))} \geq \frac{threshold}{1 + threshold} = \theta. \tag{2}$$

$$\begin{aligned} A(a \cap b) =& [\min(X2_a, X2_b) - \max(X1_a, X1_b)] \times \\ & [\min(Y2_a, Y2_b) - \max(Y1_a, Y1_b)]. \end{aligned} \tag{3}$$

The IoU computation is formulated in Eq.(1), When the IoU between these candidate bounding boxes exceeds a predefined threshold, it implies that they describe the same entity. To streamline the computation and reduce complexity, the inequality Eq.(2) derived from comparing the IoU with the threshold is transformed, where θ is represent as the transformed threshold. Guided by this inequality Eq.(2), the design of the computation units, as depicted in Fig.2, is developed. The intersection area between candidate bounding box a and candidate bounding box b can be computed using Equation Eq.(3), where $(X1_a, Y1_a, X2_a, Y2_a)$ and $(X1_b, Y1_b, X2_b, Y2_b)$ represent the respective positional information of box a and b. Their respective confidence scores is respectively represented as S_a and S_b.

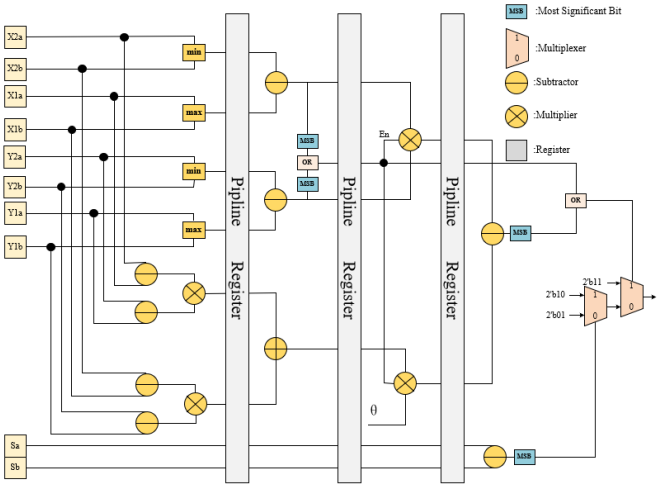


Fig. 2. The schematic of PCU

In fact, each candidate bounding box has limited area overlap with only a small subset of other candidate bounding boxes. Hence, during the computation process, when it is observed that two candidate bounding boxes have no overlap, indicated by either $[\min(X2_a, X2_b) - \max(X1_a, X1_b)]$ or $[\min(Y2_a, Y2_b) - \max(Y1_a, Y1_b)]$ being less than zero, it enables the determination that the two candidate bounding boxes do not possess any overlapping region. As a result, subsequent computations can be omitted, leading to a reduction in power consumption.

3.2 dataflow and operator design

The computation time per batch is proportional to the square of the RAM size, while the transfer time per batch is proportional to the RAM size. Hence, when the RAM size is set to a large value, the proportion of data transfer time becomes negligible. In such cases, trading more chip area for marginal speed improvement is not a favorable choice. However, we also aim to enhance data reuse and reduce the frequency of data transfers. Our NMS processor was built with an 8KB on-chip RAM that can store up to 1024 quantized candidate bounding boxes of 64 bits each. Also, we adopt the dynamic batch scheduling strategy shown in Fig.3, which leverages data flow control to minimize data transfer time.

Due to the NMS algorithm's requirement to perform pairwise computations on all candidate boxes, both inter-batch and intra-batch NMS processing are needed for each batch. To address this, as illustrated in Fig.3, we introduce two operators: $compute\ self(CS)$ for inter-batch NMS processing and $compute\ other(CO)$ for intra-batch NMS processing. In the task scheduling scheme depicted in the first column of Fig.3, the data in RAM0 remains unchanged throughout the process. After the completion of each CO operation, the

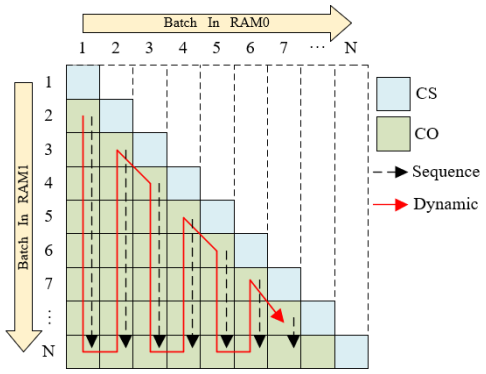


Fig. 3. Dynamic batch scheduling

data in RAM1 is updated following the dynamic scheduling curve. Once the data in RAM1 is updated to the Nth batch and the *CO* operation is accomplished, RAM1's data is preserved while updating the data in RAM0. Consequently, this dynamic scheduling approach results in the reduction of one batch of data transmission compared to the sequential scheduling demonstrated in Fig.3. Similarly, at the turning point of the second column in Fig.3, where RAM0 stores the second batch of data and RAM1 stores the third batch, a logical swap is performed between RAM0 and RAM1. This exchange allows dynamic scheduling to reduce data transmission by one batch compared to sequential scheduling. In conclusion, dynamic scheduling can reduce N-1 batches of data transmission in comparison to sequential scheduling. Furthermore, the *CS* operation can be performed when the data of each batch is first cached on-chip.

4. Experiment

In order to evaluate the performance of the proposed NMS processor architecture design, comparative analysis was conducted between the performance of the NMS processor and conventional computing units such as CPU and GPU, as well as with other state-of-the-art NMS accelerators. To ensure the validity and reliability of the comparisons, real-world road scene images were used as input data. Moreover, the input candidate bounding boxes for the NMS processor were obtained from the YOLOv5[29] object detection neural network.

Table I. comparison with General-purpose Hardware

	Accuracy	Latency
ARM Cortex-A72	100%	9 ms
AMD Ryzen 9 5950X	100%	3 ms
RTX 2080Ti	100%	2 ms
Our Work	99.6%	16.6 μs

Our performance analysis of the NMS processor, as presented in this study, was conducted utilizing the NCSU-FreePDK45 academic library. For comprehensive evaluation of the on-chip storage units' characteristics, encompassing both area and power considerations, we employed the widely acknowledged open-source Memory Compiler tool, OpenRAM[30]. Ultimately, utilizing the 45nm process technology, our NMS processor operates at a clock frequency of 500MHz, whose area is 526,752 μm^2 and power consumption is 58.681 mW. As illustrated in Table I, our proposed NMS processor exhibits significantly reduced processing time compared to advanced general-purpose processors when performing end-to-end NMS tasks. Remarkably, it mitigates excessive accuracy loss resulting from quantization and computation during end-to-end task processing. This attribute lends itself remarkably well to the efficient completion of NMS operations in intricate traffic object detection scenarios, thus positioning it as an optimal solution for NMS deployment at the edge.

Table II. comparison with previous Research

	[15]	[17]	Ours
Technology	28nm	40nm	45nm
Frequency (MHz)	400	400	500
Area (μm^2)	80360	37867	526752
Accuracy	97.7%	100%	99.6%
Latency(μs)	12.79	203.75	16.6

As illustrated in Table II, we conducted a comparative analysis between our proposed NMS accelerator and previous state-of-the-art works. In terms of processing speed, our NMS processor is comparable to the previous one, with a minimal difference in the time required for processing 1000 candidate boxes (50 target entities). However, it outperforms the previous processor by 2% in processing accuracy, significantly reducing the workload on the post-detection processor. For safety-critical tasks like autonomous driving and assisted driving, high-precision detection results are crucial to ensure informed decision-making, risk mitigation, and accident prevention.

5. Conclusion

In this brief, we present a high-accuracy and low-latency NMS processor designed to efficiently process a significant number of candidate bounding boxes generated by front-end object detection neural networks in complex traffic road scenes. Our proposed NMS processor surpasses previous approaches by delivering enhanced accuracy and effectively addressing more intricate object detection scenarios. Moreover, it eliminates the need for candidate bounding box sorting, thereby enhancing the parallelism and reducing the processing time of the end-to-end NMS task. The obtained results demonstrate the efficient edge computing capability of our work for the NMS algorithm in real-world traffic environments.

References

[1] T. Meinhardt, et al.:"TrackFormer: Multi-Object Tracking with Transformers," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR) (2022) 8834-8844 (DOI: 10.1109/CVPR52688.2022.00864).

[2] H. Wang, et al.: "CenterNet-Auto: A Multi-object Visual Detection Algorithm for Autonomous Driving Scenes Based on Improved CenterNet," IEEE Transactions on Emerging Topics in Computational Intelligence 8 (2023) 742 (DOI: 10.1109/TETCI.2023.3235381).

[3] X. Wanget,et al.: "Multi-Object Detection in Traffic Scenes Based on Improved SSD," Electronics 7 (2018) 302 (DOI: 10.3390/electronics7110302)

[4] X. Chen,et al.: "Multi-view 3D Object Detection Network for Autonomous Driving," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 6526 (DOI: 10.1109/CVPR.2017.691).

[5] K. Chitta, et al.: "TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving," IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 1 (DOI: 10.1109/TPAMI.2022.3200245).

[6] H. Wu,et al.:"Adversarial Driving: Attacking End-to-End Autonomous Driving," 2023 IEEE Intelligent Vehicles Symposium (IV) (2023) 1 (DOI: 10.1109/IV55152.2023.10186386).

[7] V. H. Kim and K. K. Choi:"A Reconfigurable CNN-Based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA," IEEE Access 11 (2023) 59438 (DOI: 10.1109/ACCESS.2023.3285279).

[8] Yan Xuet, et al.:"Design and implementation of an efficient CNN accelerator for low-cost FPGAs," IEICE Electronics Express 19 (2022) 20220370 (DOI:https://doi.org/10.1587/elex.19.20220370).

[9] Han Jia,et al.:"An FPGA-based accelerator for deep neural network with novel reconfigurable architecture," IEICE Electronics Express 18 (2021) 20210012 (DOI:https://doi.org/10.1587/elex.18.20210012).

[10] Shichang Zhang, et al.:"Thread: Towards fine-grained precision reconfiguration in variable-precision neural network accelerator," IEICE Electronics Express 16 (2019) 20190145 (DOI:https://doi.org/10.1587/elex.16.20190145)

[11] J. Lee, C. Kim, et al.: "UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision," in IEEE Journal of Solid-State Circuits 54 (2018) 173 (DOI: 10.1109/JSSC.2018.2865489).

[12] Y. -H. Chen, et al.: "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," IEEE Journal of Solid-State Circuits 52 (2017) 127 (DOI: 10.1109/JSSC.2016.2616357).

[13] Feng Liu, et al.: "CASSANN-v2: A high-performance CNN accelerator architecture with on-chip memory self-adaptive tuning," IEICE Electronics Express 19 (2022) 20220124 (DOI: https://doi.org/10.1587/elex.19.2022012465).

[14] Zhen Wang,et al.:"EERA-DNN: An energy-efficient reconfigurable architecture for DNNs with hybrid bit-width and logarithmic multiplier," IEICE Electronics Express 15 (2018) 20180212 (https://doi.org/10.1587/elex.15.20180212).

[15] M. Shi, et al.:"A Fast and Power-Efficient Hardware Architecture for Non-Maximum Suppression," IEEE Transactions on Circuits and Systems II: Express Briefs 66 (2019) 1870-1874 (DOI: 10.1109/TCSII.2019.2893527).

[16] K. Ning, et al.:"A High-Speed NMS Coprocessor for Lightweight Ship Detection Algorithm," IEEE Transactions on Circuits and Systems II: Express Briefs 69 (2022) 1677 (DOI: 10.1109/TCSII.2021.3125487).

[17] Y.Peng, et al.:"AnchorCapsule : adatastream- serving post-processor for object detection in embedded vision SoC," IEEE Transactions on Circuits and Systems II: Express Briefs (2022) 1 (DOI: 10.1109/TCSII.2022.3200038.)

[18] S. B. Choi, et al.:"Implementation of post processing hardware for real-time object detection in cnn acceleration system," 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) (2021) 1 (DOI: 10.1109/ITC-CSCC52171.2021.9501433).

[19] F. Chaoming,et al.:"A sort-less fpga-based non-maximum suppression accelerator using multi- thread computing and binary max engine for object detectionn," IEEE Asian Solid-State Circuits Conference (2021) 1 (DOI: 10.1109/ITC-CSCC52171.2021.9501433).

[20] C. Chen, et al.:"Scalable hardware acceleration of non-maximum sup- pression," Design, Automation Test in Europe Conference Exhibition (DATE) (2022) 96 (DOI: 10.23919/DATE54114.2022.9774717).

[21] T. Zhang, et al.:"Psrr- maxpoolnms: Pyramid shifted maxpoolnms with relationship recovery," IEEE/CVF Conference on Computer Vision and Pattern Recog- nition (CVPR) (2021) 15835 (DOI: 10.1109/CVPR46437.2021.01558)

[22] Z. Zhang, et al.: "Strict NMS: Pedestrian Detection in Crowd Scenes," IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE) (2020) 225 (DOI: 10.1109/ICISCAE51034.2020.9236802).

[23] Y. Tang,et al.:"OTP-NMS: Toward Optimal Threshold Prediction of NMS for Crowded Pedestrian Detection," in IEEE Transactions on Image Processing 32 (2023) 3176 (DOI: 10.1109/TIP.2023.3273853).

[24] A. Kumar, et al.:"GrooMeD-NMS: Grouped Mathematically Differentiable NMS for Monocular 3D Object Detection," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 8969 (DOI: 10.1109/CVPR46437.2021.00886).

[25] L. Tian and Z. Zhang:"Simple-NMS: Improved Pedestrian Detection with New Constraints," International Conference on Computer Communication and Artificial Intelligence (CCAI) (2021) 30 (DOI: 10.1109/CCAI50917.2021.9447459).

[26] Liu S, Huang D, Wang Y:"Adaptive nms: Refining pedestrian detection in a crowd" Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2019) 6459 (DOI: 10.1109/CVPR.2019.00662).

[27] N. Bodla, et al.:"Soft-NMS — Improving Object Detection with One Line of Code," IEEE International Conference on Computer Vision (ICCV) (2017) 5562 (DOI: 10.1109/ICCV.2017.593).

[28] Y. He, et al.:"Bounding Box Regression with Uncertainty for Accurate Object Detection," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2883 (DOI: 10.1109/CVPR.2019.00300).

[29] JOCHER,et al., "YOLOv5: v3.1 - Bug Fixes and Performance Improvements" online (2021) (DOI:http://doi.org/10.5281/zenodo.4154370) .

[30] M. R. Guthaus,et al.:"OpenRAM: An open-source memory compiler," 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (2016) 1 (DOI: 10.1145/2966986.2980098).