Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 4 (2025)
https://www.jneonatalsurg.com

OPEN ACCESS

Real-Time Pedestrian Surveillance System Based on Deep Learning Object Detecting Algorithm

Woo-Jin Jung[1], Won-hyuk Choi[*2]

[1]Department of Aeronautical System Engineering, Hanseo University, Taean 32158, Korea
[2]Department of Avionics, Hanseo University, Seosan, Taean 32158, Korea

Cite this paper as: Woo-Jin Jung, Won-hyuk Choi, (2025) Real-Time Pedestrian Surveillance System Based on Deep Learning Object Detecting Algorithm. *Journal of Neonatal Surgery*, 14 (4), 306-313.

ABSTRACT
Smart crosswalks represent a novel approach to traffic management, employing the integration of Internet of Things (IoT) sensors and real-time control of traffic signals to mitigate the risks associated with pedestrian accidents. The advent of advanced deep learning algorithms for object detection has enhanced the feasibility of developing effective real-time pedestrian detection systems. The YOLO (You Only Look Once) algorithm, renowned for its high degree of accuracy and rapid object recognition, has undergone substantial enhancements. Notably, the YOLOv8 model has demonstrated substantial advancements in performance compared to its predecessors through the integration of the C2f module, PAN module, and the adoption of an anchor-free approach. This study has established a comprehensive pedestrian dataset for the evaluation of the efficacy of various YOLO versions, and has identified the optimal algorithm version for pedestrian detection through rigorous practical training. An on-device pedestrian detection system has been developed by deploying the trained algorithm onto a Jetson Nano board, and subsequent simulation tests have been conducted to verify the accurate detection of pedestrians.

Keywords: *Pedestrian, Yolo, Deep Learning, Jetson Nano, Object Detecting*

1. INTRODUCTION

PCOS is a known illness of the endocrine system, It primarily affects 4%–20% of women who are of reproductive age (1) causing oligo-ovulation, hyperandrogenism, and polycystic ovary morphology. The number of road traffic fatalities per 100,000 children in Korea is 1.4, which is higher than the Organization for Economic Co-operation and Development (OECD) average of 1.0. Furthermore, the number of pedestrian fatalities per 100,000 children is 0.88, which is higher than the OECD average of 0.31. Furthermore, the number of road accidents involving children in Korea has increased over the past three years. In response, the National Assembly recently passed the Civil Diet Act, which aims to enhance safety in child protection zones and crosswalks, with the objective of preventing accidents.

A typical application of a pedestrian safety platform is the use of smart crosswalks, which employ IoT sensors and real-time signal control technology with the objective of reducing the risk of pedestrian accidents. The platform can provide the technical and policy foundation necessary to support the deployment and operation of smart crosswalks. An AI-powered pedestrian crossing system that is capable of detecting pedestrians in a crosswalk and communicating this information to the vehicle in question has the potential to proactively prevent traffic accidents.

The recent advancements in deep learning algorithms have enhanced the capabilities for the development of systems that are capable of real-time object detection. Representative deep learning-based object detection algorithms include SSD [1], YOLO [2] and R-CNN [3]. These algorithms necessitate a GPU-based embedded vision environment for data processing and on-board risk analysis. The Linux-based Jetson Nano Board with NVIDIA GPU is a popular choice for the construction of deep learning-based systems, given its capacity to facilitate deep learning-based image processing.

Accordingly, this paper puts forth a real-time deep learning algorithm for pedestrian detection and offers an analysis of the algorithm's performance through training. Subsequently, the deep learning algorithm is ported to the GPU-based Jetson Nano board, thereby facilitating the construction of a real-time pedestrian monitoring system.

2. SELECT OBJECT DETECTION ALGORITHM

The structural synthesis of CCPGTs will be performed based on the creative design methodology process [7-8]. Fig. 3 shows the flow chart for the approach. The process consists of six steps:

2.1 Object Detection Algorithm

To detect pedestrians, it is necessary to utilize an object detection algorithm that can be processed in real time. The process of object detection entails the identification of an object's location within a given input image, coupled with its classification. The field of deep learning-based object detection algorithms encompasses two principal categories: two-stage detectors and one-stage detectors. In two-stage detectors, the initial neural network proposes the region of the object, and the subsequent network classifies the proposed location and identifies the precise location. Notable exemplars of these computational methodologies include R-CNN, fast R-CNN [4], mask R-CNN [5], and cascade R-CNN [6]. Although these two-stage detectors demonstrate efficacy, their latency is excessive, rendering them unsuitable for real-time object detection. To address these limitations, single-stage detectors, including SSD, YOLO, and RetinaNet, have demonstrated faster object detection speeds than two-stage detectors. However, this increased speed is accompanied by a reduction in accuracy. The recent advancements in deep learning algorithms have led to improvements in accuracy. In this paper, we have constructed a system utilizing YOLO-like algorithms among one-stage detectors.

2.2 Algorithm Selection

The most prevalent YOLO-based algorithms currently in use are YOLO v5 [7], v7 [8] and v8. The YOLO model is comprised of three distinct components: the spine, the neck, and the head. YOLOv8 represents the latest iteration in the YOLO series of algorithms. YOLOv8 employs the C2f module, which represents an enhancement over the C3 module utilized in the CSPDarkNet backbone module of the YOLOv5 model. The C2f module represents a conceptual innovation introduced by the ELAN module [9], which comprises a bottleneck block incorporating two convolutional layers utilizing a 1x1 kernel. The addition of a residual concatenation method enables the computation to be performed at a faster speed than that of the existing C3 module.

The classification model generates a feature map by progressively down sampling the input image over several layers. As higher layers are incorporated, the receptive field expands while the feature map diminishes in size. The feature maps obtained at the top layer exhibit a high level of semantic information, yet the spatial information is lost and the resolution is reduced. This results in a loss of detail. To address these shortcomings, the PAN module was incorporated into the YOLOv4 [10] network as part of the neck module. The PAN [11] module performs up sampling of feature maps from higher layers and combines them with feature maps from lower layers, thereby ensuring that each layer's feature map contains both spatial and semantic information. This is referred to as a feature pyramid structure. In YOLO, the combination of feature maps is conducted on a channel-by-channel basis, as opposed to an additive approach, with the objective of conveying information in a more efficient manner. The PAN structure has been retained in YOLOv8.

In addition to enhancements to the model structure, YOLOv4 also introduced novel data augmentation techniques that led to a notable improvement in performance. Mosaic is a representative augmentation technique whereby four training images are collected and pasted in a checkerboard format, thereby enabling the model to learn simultaneously from all four images.

This has the potential to enhance the recognition rate of small objects; however, it has been demonstrated that continuous data augmentation can result in a decline in learning performance. Consequently, YOLOv8 employs data augmentation up to a scale of 10.

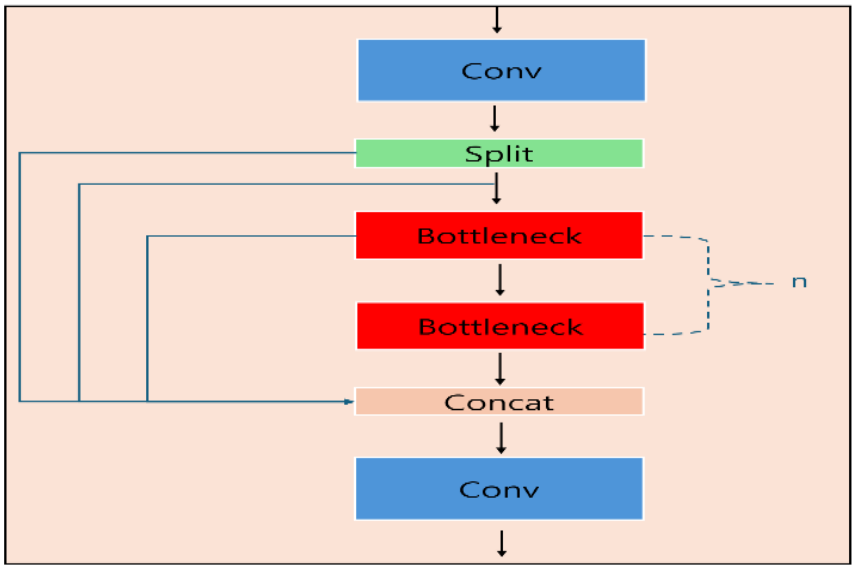

Fig. 1 C2f Module

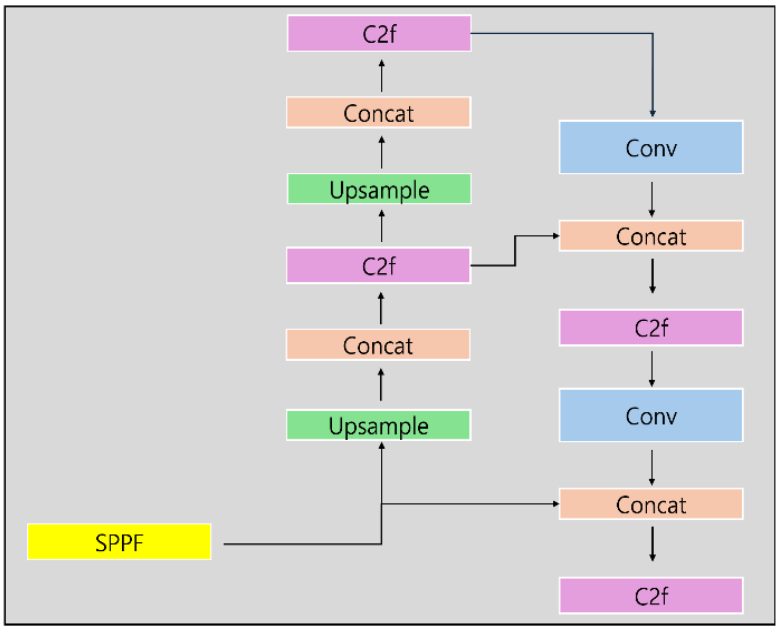

Fig. 2 YOLOv8 PAN Structure

2.3 YOLOv8 PAN Algorithm

The YOLOv8 model has been designed as an anchor-free alternative to the previous YOLOv5 model. The use of anchor boxes is eliminated, thus reducing the amount of unnecessary computation. The anchor-free method [12] employs a grid-based recognition approach in the output feature map, which reduces the amount of computation due to the reduction in the number of dimensions and the elimination of post-processing for all anchor boxes. This results in an increase in the speed of object detection, which is advantageous for real-time object detection. Table 1 illustrates the training outcomes of the most prevalent YOLO-based algorithms on the COCO dataset. It is evident that YOLOv8 exhibits superior performance in comparison to other versions of the model, even when the same series of algorithms is considered. Accordingly, this study trained algorithms for a comparative analysis of YOLOv8 and Yolov5, which exhibit enhanced capabilities for rapid object inference and precision compared to existing models. This was achieved by optimizing the structural design of existing YOLO-based algorithm models.

Model	mAP	mAP@50
v5n	28	45.7
v5s	37.4	56.8
v5m	45.4	64.1
v5l	49	67.3
v5x	50.7	68.9
v7-tiny	37.4	55.2
v7	51.2	69.7
v7x	52.9	71.1
v8n	37.3	52.5
v8s	44.9	61.8
v8m	50.2	67.2
v8l	52.9	69.8
v8x	53.9	71.0

Fig. 3 COCO Benchmarks

3. ALGORITHM TRAIN

3.1. Configure Datasets

The dataset comprised a total of 4,638 labelled custom pedestrian images. The Train, Validation, and Test datasets comprise 4,056, 397, and 185 images, respectively. An exemplar of a training image is provided below for illustrative purposes. The training data was comprised of a single class, designated as "Person."

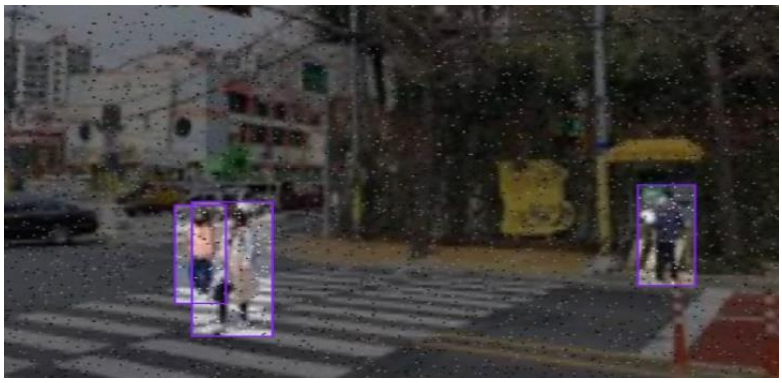

Fig. 4 Simulation Dataset

3.2. Algorithm Train

Once the Pytorch and CUDA environment had been established on the virtual environment of the RTX 3060 ti single GPU, the Yolov5 and v8 models (versions n, s, m, and l) were trained on the Jupyter laptop environment with 50 epochs and 32 batch sizes. The mean time required for training is 39 minutes, with the fastest training speed being 22 minutes for the Yolov5n and Yolov8n models.

Category	Setting
GPU	RTX 3060 ti
DATA Set	Train: 4056, Validation: 397 Test: 185
Pytorch	2.0.1
Cuda	12.6
Batch	32
Epoch	50
learning Rate	0.01

Fig. 5 Simulation Settings

3.3. Analyze Results

The primary metrics for assessing the efficacy of deep learning-based object detection algorithms are the average intersection over union (IoU), recall, precision, and mean average precision (mAP). The Jaccard index, also known as IoU, is a method of quantifying the degree of overlap between the target and the prediction. Precision can be defined as the proportion of detectors that the model identifies as true values that are, in fact, accurate. Recall can be defined as the proportion of data that the model correctly identifies as true. The formulas for precision and recall are provided below. The result is obtained by calculating the area under the curve (AUC) after calculating the precision-recall curve, which is a graph that visualises the trade-off between precision and recall for all object classes separately. The mean average precision (mAP) value for all classes is identical to the average precision (AP) value, given that only a single pedestrian class was trained in this experiment. The results for precision, recall, mAP@50, and loss for the simulation are presented in Table 3. It should be noted that the results for the initial 10 epochs are unreliable; therefore, only the results from epoch 11 onwards, when the simulation stabilises, are presented. In general, the v8 model demonstrated superior performance compared to the v5 model of the same version in terms of precision, recall, mAP@50, and loss. The m version of both the v5 and v8 models demonstrated the most

optimal results. Despite the v8s model exhibiting a higher computational speed, it exhibited similar accuracy to the v8l model.

Figures 7 and 8 illustrate the evolution of mAP and loss values over the actual epoch. Prior to 43 epochs, all models except the N model in V8 demonstrated superior performance compared to the other models. However, in the final 50 epochs, the M and L models in V5 exhibited a notable enhancement in their performance. Nevertheless, this outcome remained inferior to that of the M model in V8. With regard to the loss function, the s, m, and l models of the v8 exhibited superior performance compared to the other models across the entire range of epochs, with the v8 l model demonstrating the highest performance at 0.487. The V8 M model also exhibits a commendable performance of 0.489, exhibiting a mere 0.002 difference from the maximum performance of the V8 L model. Consequently, it is the most optimal in terms of overall performance metrics, with the exception of Loss. The V8 N model, which exhibits the second highest Loss result, is selected as the algorithm for the construction of a pedestrian recognition system. Figure 9 illustrates the test results of the actual Yolov8m model, demonstrating that the pedestrian is correctly identified with a high result of 0.9.

$$Reproducibility = \frac{TP}{FN+TP} \quad (1)$$

$$Precision = \frac{TP}{FP+TP} \quad (2)$$

Model	Precision	Recall	mAP@50	Loss_Min
v5n	0.847	0.862	0.910	0.623
v8n	0.872	0.876	0.928	0.584
v5s	0.860	0.872	0.922	0.540
v8s	0.888	0.894	0.940	0.517
v5m	0.876	0.878	0.931	0.514
v8m	0.893	0.899	0.946	0.489
v5l	0.866	0.873	0.921	0.512
v8l	0.891	0.890	0.937	0.487

Fig. 6 Simulation Result

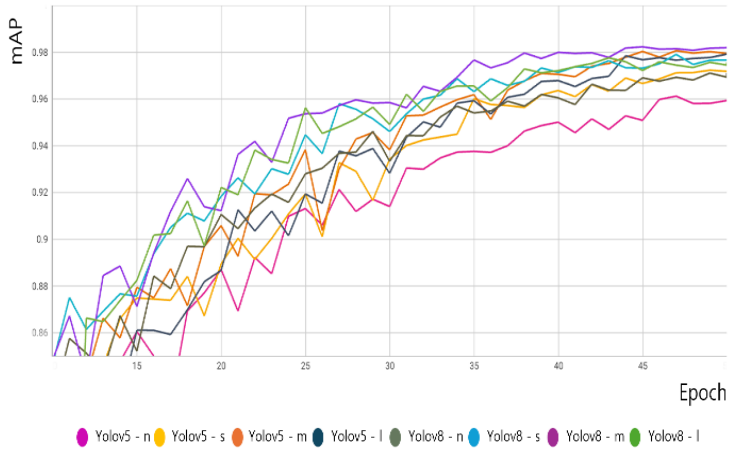

Fig. 7 mAP Results

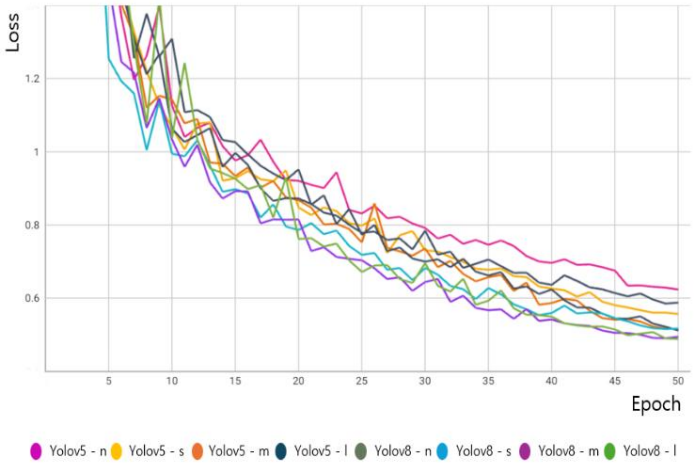

Fig. 8 Loss Results

Fig. 9 Yolov8m Training Results

4. CONCLUSIONS PEDESTRIAN DETECTION SYSTEM BASED ON JETSON NANO

The system proposed in this paper employs a deep learning object detection algorithm, trained on a GPU-based Jetson Nano board equipped with a camera, to detect pedestrians. As the system is based on an image analysis model, the use of SWAP memory [13] serves to enhance the system's stability. The NVIDIA TensorRT [14] deep learning inference application was employed to facilitate low latency and high throughput for real-time systems. YOLOv8 employs PyTorch for both training and inference, which can be transformed into TensorRT models for inference in order to achieve enhanced computational performance. TensorRT is the most effective of the CUDA frameworks for inference computation. However, as the PyTorch model cannot be converted to a TensorRT model immediately, we applied the TensorRT application by converting the ONNX model [15] in order to enhance the reliability of the object verification process in the real-time system. The actual system implementation is illustrated in Figure 10. By establishing a connection between the camera linked to the Jetson Nano via USB and the Jetson Nano itself, the previously trained YOLOv8 algorithm is linked to the camera calling program, thereby enabling the extraction of pedestrians. The system's flowchart is illustrated in Figure 10.

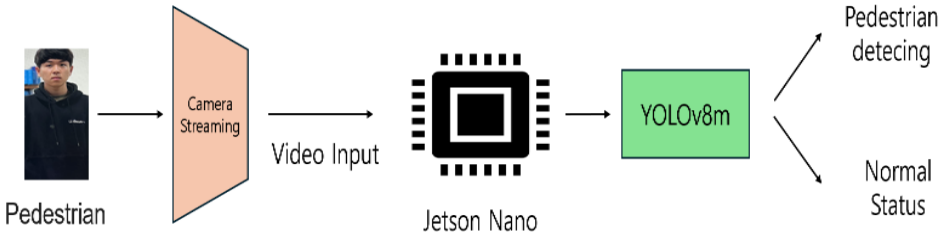

Fig. 10 System Diagram

In accordance with the aforementioned system flowchart, the actual system environment is represented by Figure 11.

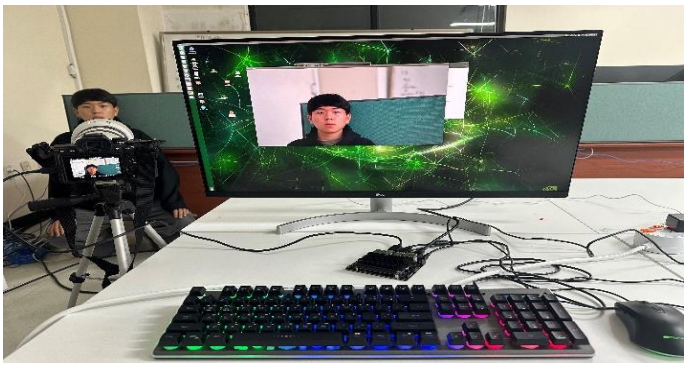

Fig. 11 System Settings

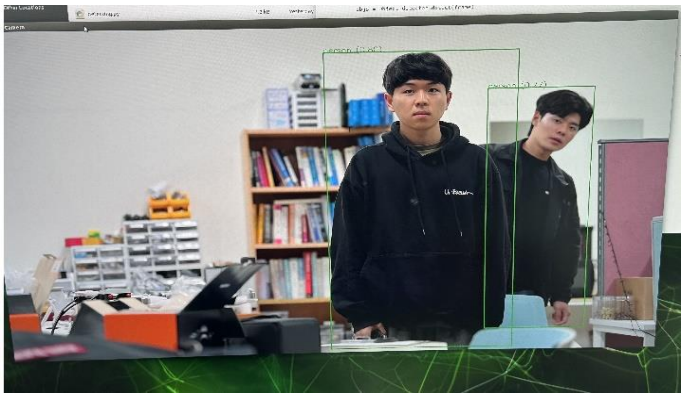

Fig. 12 System Results

Following the configuration of a program to run YOLOv8 on the Jetson Nano operating system with pre-trained pedestrian objects, the actual results of pedestrian detection are presented in Figure 12. It can be observed that both moving and stationary pedestrians are correctly identified.

5. CONCLUSIONS

In this study, the utilization of Internet of Things (IoT) sensors and real-time signal control technology in the context of smart pedestrian crossings represents a pioneering technological advancement with the potential to mitigate the risk of pedestrian accidents. The advancement of deep learning algorithms has facilitated the development of real-time pedestrian detection systems. The Yolo algorithm is a one-stage object detection algorithm, which is suitable for real-time pedestrian detection systems. The Yolov8 version enhances the recognition rate in comparison to existing models through the utilization of C2f and PAN structures, while adopting an anchor-free methodology to facilitate enhanced recognition speeds. Upon training on authentic pedestrian datasets, Yolov8 demonstrated superior performance compared to existing models. By porting the Yolov8m model, which demonstrated the best performance, to the Jetson Nano OS environment, we constructed a system that can perform real-time pedestrian detection by analyzing the input video images from cameras through deep learning. The simulation results for the on-device-based real-time pedestrian detection system demonstrated that both moving and stationary pedestrians were correctly identified. As the algorithm was trained on a single class of pedestrians for pedestrian detection, future research will focus on enhancing its performance by expanding the range of training data and situations it can recognize.

6. ACKNOWLEDGMENTS

Following are results of a study on the "Leaders in INdustry-university Cooperation 3.0" Project, supported by the Ministry of Education and National Research Foundation of Korea

REFERENCES

[1] Liu, Wei, et al, “SSD: Single Shot Multbox Detector.” in 14th European Conference, Amsterdam: The Netherlands, pp. 11-14, Sep. 2016. DOI: https://doi.org/10.1007/978-3-319-46448-0_2.

[2] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV: USA, pp. 779-788, June 2016. DOI: 10.1109/CVPR.2016.91.

[3] Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH: USA, pp. 580-587, June 2014. DOI: 10.1109/CVPR.2014.81.

[4] Girshick, R., "Fast R-CNN," [Internet]. Available: https://arxiv.org/abs/1504.08083.

[5] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, pp. 2961-2969, Oct. 2017. DOI: 10.1109/ICCV.2017.322.

[6] Cai, Z., and Vasconcelos, N., "Cascade R-CNN: Delving into High Quality Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT: USA, pp. 6154-6162, June 2018. DOI: 10.1109/CVPR.2018.00644.

[7] Jocher, G., Chaurasia, A., and Qiu, J., "Ultralytics YOLOv5," [Internet]. Available: https://github.com/ultralytics/ultralytics.

[8] Wang, C., Bochkovskiy, A., and Liao, H. Y. M., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," [Internet]. Available: https://arxiv.org/abs/2207.02696.

[9] Wang, C. Y., Yan, H., and Liao, H. Y. M., "YOLOv8: A New State-of-the-Art for Real-Time Object Detection," [Internet]. Available: https://arxiv.org/abs/2307.02444.

[10] Zhang, X., Tian, Y., Kong, Y., and Zhong, B., "Efficient Long-Range Attention Network for Image Super-Resolution," in Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, pp. 379-395, Aug. 2020. DOI: 10.1007/978-3-030-58539-6_23.

[11] Bochkovskiy, A., Wang, C. Y., and Liao, H. Y. M., "YOLOv4: Optimal speed and accuracy of object detection," [Internet]. Available: https://arxiv.org/abs/2004.10934.

[12] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S., "Feature Pyramid Networks for Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI: USA, pp. 2117-2125, July 2017. DOI: 10.1109/CVPR.2017.106.

[13] Law, H., and Deng, J., "CornerNet: Detecting Objects as Paired Keypoints," in Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, pp. 734-750, Sept. 2018. DOI: 10.1007/978-3-030-01234-2_45.

[14] NVIDIA, "NVIDIA Jetson Nano Developer Kit User Guide," [Online]. Available: https://developer.nvidia.com/embedded/learn/getting-started-jetson-nano-devkit.

[15] NVIDIA, "TensorRT Developer Guide," [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/archives/index.html.

[16] [16] ONNX, "ONNX: Open Neural Network Exchange," [Internet]. Available: https://onnx.ai/.