Deep neural network-based physical distancing monitoring system with tensorRT optimization

Edi Kurniawan [a,b,1,*], Hendra Adinanta [c,2], Suryadi [a,3], Bernadus Herdi Sirenden [d,4], Rini Khamimatul Ula [a,5], Hari Pratomo [a,6], Purwowibowo [a,7], Jalu Ahmad Prakosa [a,8]

[a] Research Center for Photonics, National Research and Innovation Agency (BRIN), Tangerang Selatan, Indonesia
[b] Magister of Information and Technology, Digital Technology University of Indonesia (UTDI), Yogyakarta, Indonesia
[c] Research Center for Hydrodinamics Technology, National Research and Innovation Agency (BRIN), Surabaya, Indonesia
[d] Research Center for Electronics, National Research and Innovation Agency (BRIN), Tangerang Selatan, Indonesia
[1] suhu.kurniawan@gmail.com; [2] hendra.adinanta@brin.go.id; [3] sury011@brin.go.id; [4] bernadus.herdi.sirenden@brin.go.id; [5] rini.khamimatul.ula@brin.go.id; [6] hari.pratomo@brin.go.id; [7] purwowibowo@brin.go.id; [8] jalu.ahmad.prakosa@brin.go.id
* corresponding author

ARTICLE INFO

Article history
Received April 16, 2022
Revised June 29, 2022
Accepted July 1, 2022
Available online July 31, 2022

Keywords
Physical distancing
Object detection
Deep learning
You only look once
TensorRT

ABSTRACT

During the COVID-19 pandemic, physical distancing (PD) is highly recommended to stop the transmission of the virus. PD practices are challenging due to humans' nature as social creatures and the difficulty in estimating the distance from other people. Therefore, some technological aspects are required to monitor PD practices, where one of them is computer vision-based approach. Hence, deep learning-based computer vision is utilized to automatically detect human objects in the video surveillance. In this work, we focus on the performance study of deep learning-based object detector with Tensor RT optimization for the application of physical distancing monitoring system. Deep learning-based object detection is employed to discover people in the crowd. Once the objects have been detected, then the distances between objects can be calculated to determine whether those objects violate physical distancing or not. This work presents the physical distancing monitoring system using a deep neural network. The optimization process is based on TensorRT executed on Graphical Processing Unit (GPU) and Computer Unified Device Architecture (CUDA) platform. This research evaluates the inferencing speed of the well-known object detection model You-Only-Look-Once (YOLO) run on two different Artificial Intelligence (AI) machines. Two different systems-based on Jetson platform are developed as portable devices functioning as PD monitoring stations. The results show that the inferencing speed in regard to Frame-Per-Second (FPS) increases up to 9 times of the non-optimized ones, while maintaining the detection accuracies.

1. Introduction

Physical distancing (PD) is one of the popular practices to prevent the transmission of COVID-19, which WHO currently endorses. The term PD refers to the minimum distance between two people to interact. The minimum distance recommended by WHO and CDC is 6 feet (2 meters) [1]–[4]. A study by Kuitunen *et al.* [5] showed that PD declined the pediatric emergency room patient volume and altered the patient traits to fewer stays yielded by respiratory diseases in Finland. Although PD has played an important role in tackling COVID-19, its implementation faces obstacles. As reported by Masters *et al.* [4], millennials in the USA had a more increased perceived threat of infection but did not rehearse as

many PD behaviors, potentially due to other obstacles such as employment, childcare, housing insecurity, or a lack of knowledge of the necessity of PD [4].

PD practices are challenging due to humans' nature as social creatures and the difficulty in estimating the distance from other people. Therefore, technology is required to overcome these challenges. One of the technologies that can be used is deep learning based computer vision (CV) [6], [7]. Deep learning, sometimes referred to as Deep Neural Network (DNN), which is a subset of Machine Learning (ML) that can be used to detect people objects in an image. The methods to detect people objects by using DNN can be divided into two approaches. The first is region-based, and the second is unified-based techniques. The first is also called a double stage detector since it has two stages to detect objects: region proposal and processing. Algorithms such as Fast-RCNN, Faster RCNN, and Mask-RCNN are classified into this method. The region-based method has high accuracy, but it features high complexity, so it is unsuitable for devices with limited computing capacities. The second method is unified-based, which is one stage technique. Thus, it is sometimes referred to as a single-stage detector. This method maps the image pixels into bounding box grid and class probabilities to detect the object. Algorithms such as YOLO and SSD are prime examples of this approach [7].

The popularity of DNN inspired many researchers to develop DNN-based algorithms for PD monitoring. Saponara *et.al.* [8] implemented YOLOv2 on Jetson Nano for PD monitoring with the video taken using the thermal camera. The experiment results showed that the method was faster compared to another method that used YOLOv3. Suryadi *et.al* [9] likened the performance of YOLOv3, YOLOv3-Tiny and MobilenetSSD on GPU to execute PD monitoring. The outcomes demonstrated that YOLOv3 suggested the most suitable detection accuratenesses corresponded to the other two techniques. Despite that, the YOLOv3-Tiny performed an enormous FPS rate. This is due to YOLOv3-Tiny is a lightweight version of YOLOv3. Rezai and Azami [10] developed DeepSocial that used YOLOv4 as the DNN model. In [10], YOLOv4 was combined with Hungarian algorithm and Kalman filter to monitor the distances among people. The use of YOLOv4 was then followed by [11], which experimented on YOLOV4 for PD monitoring under low light conditions.

PD monitoring system based on deep learning should consider two aspects, i.e., accuracy and detection speed. Accuracy depends on the chosen DNN model, while detection speed relates to the used hardware and optimization method. The graphical processing unit (GPU) offers the best computing capabilities due to performing parallel computation and ease for the model implementation compared to DSP or FPGA. While, TensorRT is one of the technologies to optimize neural network models instructed in all-powerful frameworks, calibrate for more inferior precision with excellent accuracy, and distribute to hyperscale data centers. TensorRT utilizes parallel computing ability of CUDA (Compute Unified Device Architecture) and cuDNN (CUDA Deep Neural Network). It can be implemented on embedded platforms such as jetson boards [12].

Several research works attempted to apply TensorRT optimization for DNN-based object detections can be found in [13]–[22]. Shin *et.al*, in [13], presents a performance inference method that combines the Jetson monitoring tool with TensorFlow and TRT source code on the Nvidia Jetson AGX Xavier platform. The survey of various works that assess and optimize deep learning model applications on the mobile and embedded platforms has been conducted in [14], [15]. Osipov et.al, in [16], modified the Agrifac HEXX TRAXX harvester by installing a 24 fps video camera and a single-board computer using a Canny edge detector and excess green minus excess red (ExGR) method with Otsu's binarization. A review of the study and application of deep learning based on structural damage detection was done by

Zhang *et.al*, in [17]. Kim *et. al* [18], implemented a deep convolutional neural network-based damage locating (DCNN-DL) approach to classify the steel frame images supplied as information as damaged and undamaged. Utilization of deep learning methods to support the development of driving safety on embedded platform can be found in [19]–[22]. Wu *et.al*, in [23], proposed a novel real-time infrared pedestrian detection algorithm that uses RepVGG to reconstruct the YOLOv4 backbone network. In other works, Wu *et.al*, in [24], proposed an embedded vehicle detection algorithm using Raspberry Pi 4B and Inter NCS2 neural computing stick has been improved in many aspects. TensorRT-based object detection deployed on limited on-board resources of Autonomous Mobile Robots (AMR) was developed in [25].

In this paper, we evaluate pre-trained DNN model's performance with TensorRT optimization used for PD monitoring system. The evaluated models are YOLOv3 and YOLOv4 with various resolutions running on two different machines, i.e., Jetson Xavier AGX and Jetson Xavier NX. The main contributions of this article are outlined as follow: (i) a detail configuration of PD monitoring system based on DNN and TensorRT Optimization is presented, (ii) the performance of PD monitoring system with various models of YOLO-based DNN in terms of FPS and detection accuracies is evaluated, (iii) two portable PD monitoring stations with AI computing capabilities are developed. In addition, the computing capabilities can also be adjusted based on the available power and CPU cores.

The remains of this paper is structured as follows; Section 2 presents the proposed method covering a configuration of the PD monitoring system, the YOLO model for object detection, and TensorRT-based optimization. Section 3 provides the experimental setup describing the experimental procedure, including hardware and software specifications. Results and discussion are given in Section 4, and Section 5 concludes the research work.

2. Method

2.1. Abbreviation

Throughout this paper, the complete list of abbreviations is provided in Table 1.

Table 1. List of Abbreviations

Abbreviation	Full Description
AI	Artificial Intelligence
CDC	Centers for Disease Control and Preventions
CNN	Convolutional Neural Network
COVID-19	Coronavirus Disease 2019
CSPNet	Cross Stage Partial Network
CSPDarknet53	Cross Stage Partial Darknet53
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
CV	Computer Vision
DNN	Deep Neural Network
FPN	Feature Pyramid Networks
FPS	Frame Per Second
GPU	Graphics Processing Unit
IOU	Intersection Over Union

Table 1. (cont.)

Abbreviation	Full Description
MAE	Mean of Absolute Error
MAP	Mean of Average Precision
PAN	Path Aggregation Network
PD	Physical Distancing
RCNN	Regional CNN
SPP	Spatial Pyramid Pooling
SSD	Single Shot Detector
TOF	Time of Flight
WHO	World Health Organization
YOLO	You Only Look Once
YOLOvi	YOLO i-th Version
YOLOvi-288/416/608	YOLO i-th Version with $288 \times 288/416 \times 416/608 \times 608$ input resolution

2.2. PD Monitoring

A block diagram of the PD monitoring system based on a deep learning algorithm is shown in Fig. 1. Early works on PD monitoring system based on deep learning have been done in [9], [26]. Here, we specifically used DNN-based object detector to detect people objects. A pre-trained YOLO model inferred with TensorRT optimization is used in the implementation. When the people things are caught along with their bounding containers and centroids, the next step calculates the pairwise distances among all detected objects. The calculated distance is norm distance formulated as follows:

$$\hat{d}_{1,2} = \sqrt{(c_{x1} - c_{x2})^2 + (c_{y1} - c_{y2})^2} \quad (1)$$

where $\hat{d}_{1,2}$ is the distance between object 1 and 2, (c_{x1}, c_{y1}) is the centroid coordinate of object 1, and (c_{x2}, c_{y2}) is the centroid coordinate of object 2.

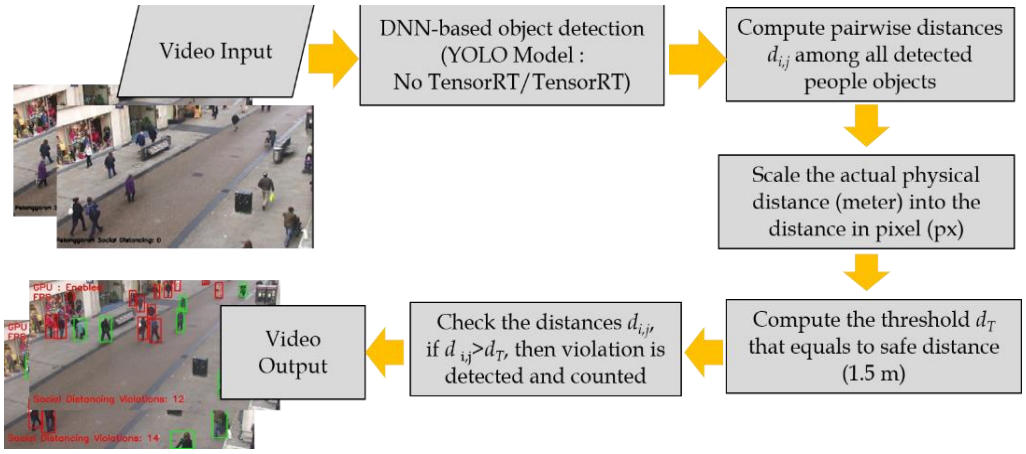

Fig. 1. Block Diagram of DNN-based PD Monitoring System

The distance $\hat{d}_{1,2}$ above is in pixel (px), and we need to know how long it represents to actual distance $d_{1,2}$. Therefore, we need a conversion from the distance in pixel to the actual distance in meter (m). The conversion can be done by following the steps as follow:

1. Obtain the focal length of camera (F)

$$F = \frac{D \times P}{W} \quad (2)$$

, where F is focal length of camera, D is the exact distance of the camera to the reference object (m), W is an actual width of reference object (m), and P is the width of reference object in an image (px). This stage is calibration step to obtain focal length of camera.

2. Estimate the actual distance (m) by using the obtained focal length of camera. The estimation formula is given by;

$$d_{1,2} = \frac{D \times \hat{d}_{1,2}}{F} \quad (3)$$

The threshold in pixel representing 1.5 m distance, can be obtained from:

$$\hat{d}_T = \frac{F \times 1{,}5}{D} \quad (4)$$

At the final stage, we check if $\hat{d}_{1,2} > \hat{d}_T$ or $\hat{d}_{1,2} < \hat{d}_T$. The violation is detected when $\hat{d}_{1,2} < \hat{d}_T$, and vice versa.

2.3. YOLO Model

Here, an actual object detection algorithm for videos based on the YOLO network proposed by Lu *et al.* [27], the Fast YOLO model has been trained for object detection to acquire the object data. YOLO splits the intake image into S x S grid, and then a single convolutional network simultaneously indicates multiple bounding boxes and class probabilities for those boxes. The YOLO detection system can be seen in Fig. 2. YOLO instructs on total pictures and directly optimizes detection performance. This unified model has several advantages over conventional methods of object detection.

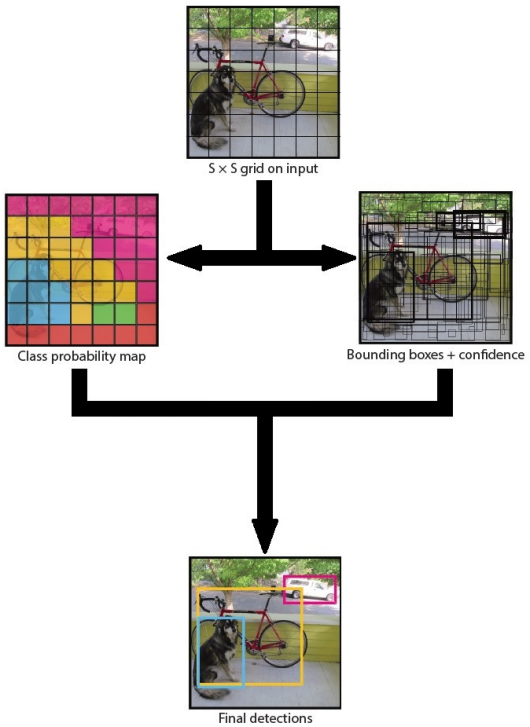

Fig. 2. YOLO detection system

There are two main reasons YOLO can perform fast and real-time detection of full images compared to other Fast R-CNN models. First, YOLO's pipeline channel just forecasts less than 100 bounding boxes per photo, while Fast R-CNN utilizes selective search. Second, YOLO performs object detection as a regression issue, so a unified architecture can directly extract attributes from information pictures to forecast bounding boxes and class probabilities [28]. YOLO's result is bounding boxes, in which each bounding box consists of 5 projections: x,y,w,h, and confidence. The (x,y) coordinates describe the center of the container. The width w and height h are predicted relative to the entire image. The confidence prediction illustrates the IOU between the indicated box and any ground truth box. YOLO recognizes an object based on class-specific.

Since its first appearance, YOLO has continued to develop. The third version of YOLO, which is YOLOv3, developed by Zao *et al*. in 2020 [29]. They have produced a new cluster strategy for calculating the initial width and height of the expected bounding boxes. A pair of width and height values are randomly selected as one initial cluster center distinct from the width and height of the ground truth boxes. Then, it creates Markov chains based on the selected initial cluster and utilizes the last points of every Markov chain as the different initial centers.

The latest version of YOLO, namely YOLOv4, was modified by Roy *et al*. in 2022 [30]. The YOLOv4 algorithm was modified with the network architecture to maximizes accuracy and rate by containing DenseNet in the backbone. It can optimize feature transfer and reuse two new residual blocks in the backbone and neck to enhance feature extraction, decreasing computing costs. The Spatial Pyramid Pooling (SPP) was used to improve the receptive field, and a modified Path Aggregation Network (PANet) maintains fine-grain localized data and improves quality fusion.

YOLO models which will be compared in this paper are YOLOv3 and YOLOv4. These choices are based upon their advantages in the accuracy of detection compared with early YOLO models. The structure of YOLO V3 and YOLO V4 are shown in Fig. 3 and Fig. 4.

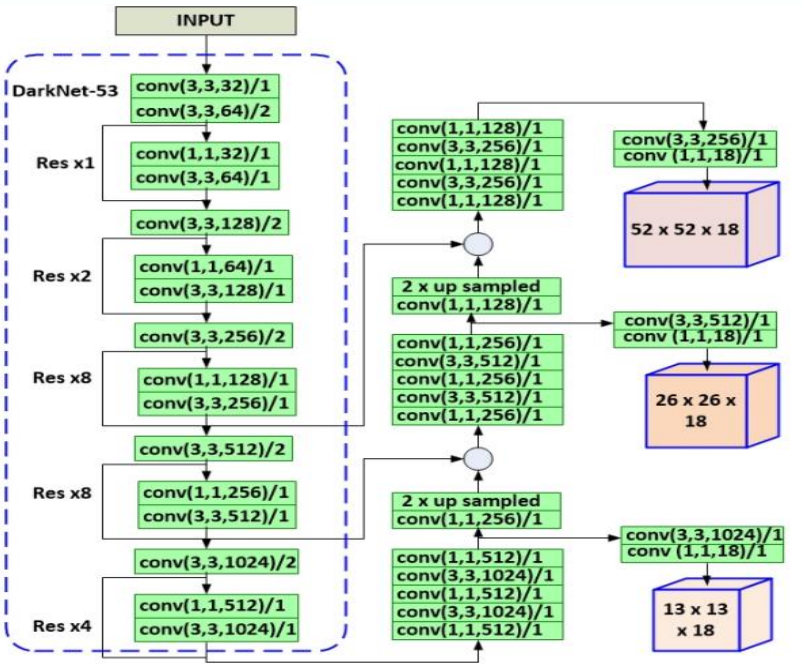

Fig. 3. YOLO V3 Structure [31]

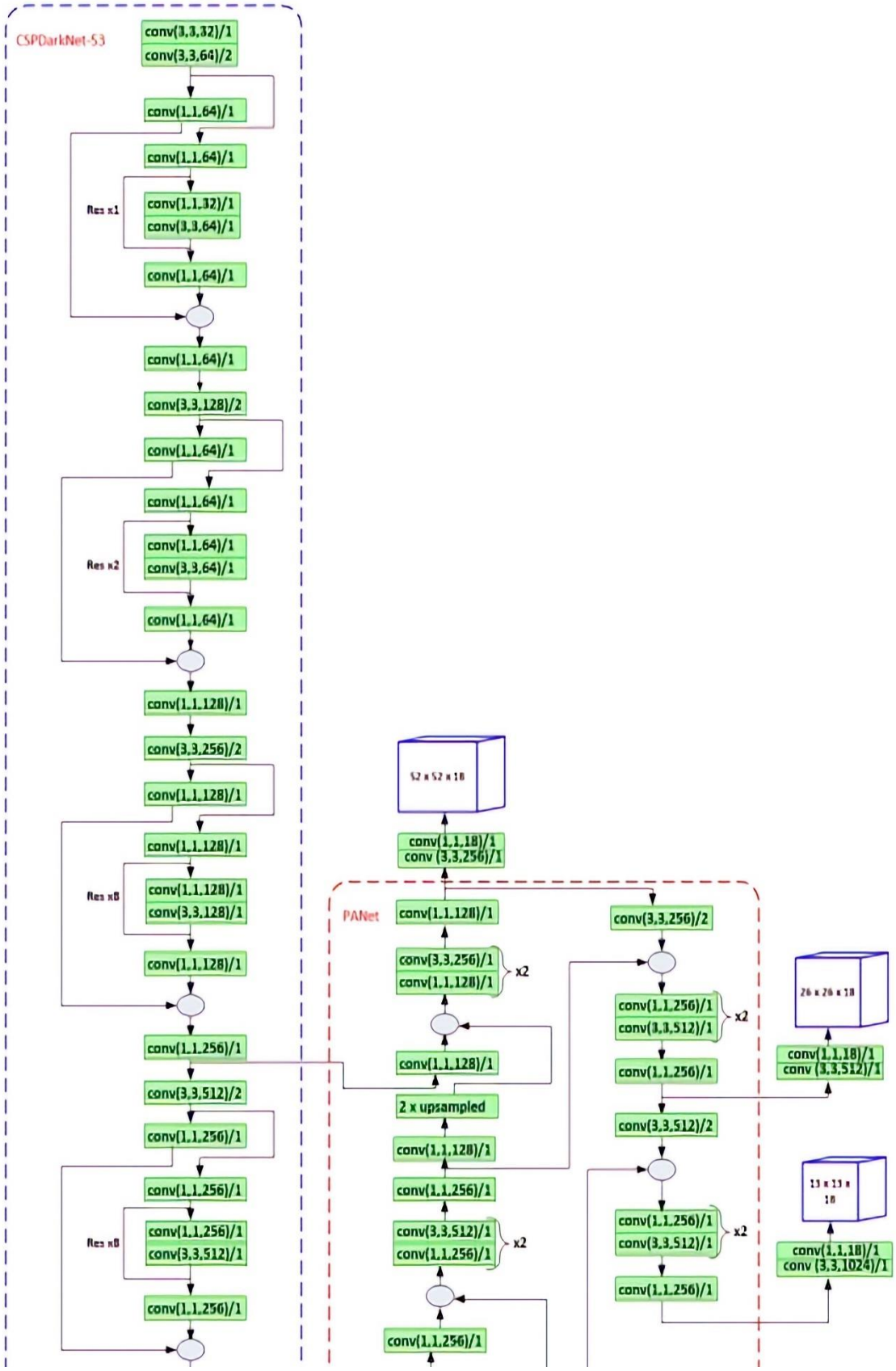

Fig. 4. YOLO V4 Structure [31]

2.4. TensorRT Optimization

In this work, TensorRT is employed to optimize the inferencing process of deep learning model. TensorRT is built on CUDA enabling inference optimization of all deep learning frameworks that utilize libraries, development tools, artificial intelligence, high-performance computing, and graphics. TensorRT delivers INT8 and FP16 optimizations for deep learning inference production applications such as video streaming, speech recognition, recommendations, fraud detection, and natural language recommendations. Smaller precision inference considerably declines application latency, requiring much real-time assistance, automated applications, and embedded applications.

Number representation is very useful when improving latency efficiency of computation system. Most of the already-trained networks use FP32 (Floating Point 32) to adjust their weight and activation function. When this parameter used in inference, it's required large memory allocation and could lead to high latency in computing process. TensorRT have the capability to convert FP32 number to lower precision number such as FP16 (Floating Point 16) or INT8 (Integer 8). This conversion can reduce computation time. The equation below formulates the process of quantization to INT8 where input, floating point range, and scaling factor are denoted by x, r, and s respectively [32].

$$Quantize\ (x,r) = round(r * clip(x, -r, r) \tag{5}$$

To maintain accuracy after conversion, TensorRT then provides calibration mechanism. This mechanism produces a set of representative data (called as calibration data), which gives histogram activation values used to find the threshold for minimum KL-divergence. TensorRT uses KL-divergence to measure the difference between the original and the new conversion numbers [33].

2.5. Experimental Setup

This research work is carried out on two PD monitoring systems, in which each system is powered by two different AI computers, namely Jetson Xavier AGX and Jetson Xavier NX. Fig. 5 and Fig. 6 show the portable PD monitoring stations which have been developed. The main hardware and software specifications are listed in Table 2. For the testing purposes, we use a recorded video of pedestrians in a busy downtown area in Oxford [34]. The supplied video is in the mp4 format consisting of 530 frames with the original frame size of 1980 x 1080 pixels.

Fig. 5. Portable PD monitoring station with Jetson AGX Xavier

Fig. 6. Portable PD monitoring station with Jetson Xavier NX

Table 2. Main hardware and software specifications

Item	System-1	System-2
GPU	384 NVIDIA CUDA® cores and 48 Tensor cores	512-core Volta GPU with Tensor Cores
CPU	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6 MB L2 + 4 MB L3	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
Memory	8 GB 128-bit LPDDR4x @ 51.2GB/s	32GB 256-Bit LPDDR4x | 137GB/s
Storage	250 GB SSD M2 NVME	250 GB SSD M2 NVME
Power	10 W	30 W
OS	Ubuntu 18.0.4	Ubuntu 18.0.4
JetPack	4.4 [L4T 32.4.3]	4.4 [L4T 32.4.3]
CUDA	10.2.89	10.2.89
TensorRT	7.1.3.0	7.1.3.0
cuDNN	8.0.0.180	8.0.0.180
OpenCV	4.4.0	4.4.0

3. Results and Discussion

The speed performance of PD monitoring system with Jetson AGX Xavier is plotted in Fig. 7. For the comparison purposes, six DNN models, i.e., YOLOv3-288, YOLOv3-416, YOLOv3-608, YOLOv4-288, YOLOv4-416, YOLOv4-608, are inferred with and without TensorRT optimization. To know what YOLOv3-288, YOLOv3-416, YOLOv3-608, YOLOv4-288, YOLOv4-416, YOLOv4-608 correspond to, please refer to the Table 1. The DNN models are executed on CUDA along with OpenCV at MAXN power mode. The MAXN power mode offers maximum utilization of both CPU and GPU cores (i.e., Online CPU: 8, Max CPU Freq.: 2265.6 MHz, GPU TPC: 4 clusters, Max GPU Freq.: 1377 MHz). When the DNN models are executed without TensorRT optimization, the maximum FPS is achieved at 2.53 for v3-288 model, and the lowest FPS is obtained for v4-608 model at 2.45. In contrast to the above results, with TensorRT optimization, the speed performances are drastically increased for all DNN models. The highest FPS is achieved for v3-288 model at 25.04, followed by v4-288 model at 24.22. The lowest FPS is obtained from v4-608 model at 13.25. The maximum percentage increase occurs on v3-288 model with 889 % improvement which is almost 9 times of the non-optimized ones. The minimum improvement is acquired on v4-608 model at 440 %. These results show drastic improvement in term of FPS when the TensorRT optimization is applied. The results also indicates that v3 models consistently achieve higher FPS compared to v4 models.

The speed performance of PD monitoring system with Jetson Xavier NX executed with and without TensorRT can be observed in Fig. 8. The DNN models are also run at maximum power mode that utilizes peak capabilities of both CPU and GPU (i.e, power budget: 15 W, online CPU: 6 cores, Max CPU Freq.: 1400 MHz, GPU TPC : 3 clusters, Max GPU Freq. : 1100 MHz). As indicated from Fig. 9, when the DNN models are inferred without TensorRT optimization, the achieved FPSs are extremely low and ranged from 1.48 to 1.71. Similar trends are shown from NX system, where the TensorRT optimization significantly increases the FPS for all DNN models. The highest FPS reaches 14.14 which is obtained from v3-288 model. This gives 849 % improvement which is relatively close to the percentage increase obtained from AGX system. The lowest FPS is also acquired from v4-608 model that is at 7.44 FPS. This contributes the minimum percentage increase by 373 %. The performance results with TensorRT optimization also point out that v3 models provide better FPS compared to v4 models.

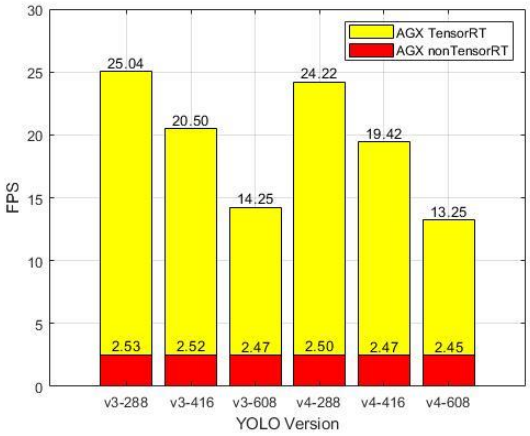

Fig. 7. Speed performance of PD monitoring system with Jetson AGX Xavier

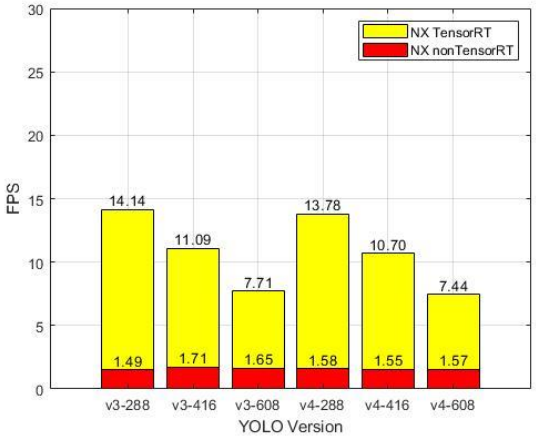

Fig. 8. Speed performance of PD monitoring system with Jetson Xavier NX

Next, we examine the speed and detection accuracy of PD monitoring systems with a static video input. The video input is created from 100-th frame of the previous video as illustrated in Fig. 9. Then, the frame is looped for 530 times. In this way, the number of persons for each frame remains the same. The ground truth of the number of persons is manually counted and obtained as 17.

Fig. 9. An image of frame # 100

Table 3 summarizes the performances in speed and detection accuracy of AGX system with TensorRT optimization. As observed in Table. 2, the similar pattern of FPS is shown here, where v3-288 model achieves highest FPS and v4-608 model offers lowest FPS. Table 3 describes that v3 models show inconsistent detection error where the model with highest resolution 608×608 results in largest detection error. The detection error (DE) here is defined as

$$DE = \frac{|GT-DP|}{GT} \text{x } 100\% \quad (6)$$

where GT and DP stand for Ground Truth and Detected Persons. In contrast to v3 models, v4 models provide consistent outcomes, where increasing the model resolution improves the detection accuracy.

The best detection accuracy is obtained from v4-608 model with 0.12 % detection error. In exchange to the best detection accuracy by v4-608 model, is the lowest processing speed which is leveled at 13.07 FPS.

Table 3. Speed and accuracy of AGX system with TensorRT optimization using 100-th frame video input

No	YOLO Model	Grund Truth #Person	TensorRT		
			#FPS	*#Avg. Detected Persons*	*%Detection Error*
1	v3-228	17	24.05	18.00	5.88
2	v3-416	17	19.54	16.97	0.17
3	v3-608	17	13.81	19.02	11.88
4	v3-228	17	23.11	14.73	13.35
5	v3-416	17	18.53	16.00	5.88
6	v3-608	17	13.07	17.02	0.12

The speed and detection performance of NX system with TensorRT optimization is described in Table 4. Similar trends to AGX system are shown from Table 4, where the fastest FPS is achieved by v3-288 model, and v4-608 model offers the lowest FPS. Table 4 also points out that the best detection accuracy is yielded by v4-608 model with 0.12 % detection error. These results confirm that increasing the model resolution improves the detection accuracy.

Table 4. Speed and accuracy of NX system with TensorRT optimization using 100-th frame video input

No	YOLO Model	Grund Truth #Person	TensorRT		
			#FPS	*#Avg. Detected Persons*	*%Detection Error*
1	v3-228	17	15.29	18.98	11.64
2	v3-416	17	11.82	21.00	23.52
3	v3-608	17	8.37	20.00	17.64
4	v3-228	17	15.15	14.73	13.35
5	v3-416	17	11.58	16.00	5.88
6	v3-608	17	7.82	17.02	0.12

4. Conclusion

The PD monitoring systems based on DNN with TensorRT optimization have been presented in this paper. The systems are built on portable workstations powered by low-power AI machines, namely Jetson AGX Xavier and Jetson Xavier NX. The TensorT optimization is executed on GPU with Computer Unified Device Architecture (CUDA) platform. In this work, the performances of DNN models with and without TensorRT optimization are evaluated. The evaluated models are YOLOv3-288, YOLOv3-416, YOLOv3-608, YOLOv4-288, YOLOv4-416, YOLOv4-608. The results indicate

that the optimization drastically improves the performance in terms of FPS on both machines. The PD station with Jetson AGX machine offers higher FPS compared to the ones with NX machine. From the experiment, it is also shown that the YOLOv4 model with highest resolution offers better accuracy in the exchange of slower computational speed. Note that the developed system is prepared for the GPU-supported machines that gives the limitation of the system. Therefore, in the future research work, a light version of this system would be developed.

Acknowledgment

The authors would like to thank the Research Center for Physics and Research Center for Photonics, National Research and Innovation Agency (BRIN) for supporting this research work. We would also like to acknowledge the Digital Technology University of Indonesia (UTDI) for making this publication possible.

Declarations

Author contribution. Edi Kurniawan: Funding acquisition, Writing-original draft, Conceptualization, Hendra Adinanta: Software, Hardware Setup, Visualization, Suryadi: Resources, Software, Hardware setup, Bernadus Herdi Sirenden: Literature, Writing-review, Rini Khamimatul Ula: Data acquisition, Project administration, Hari Pratomo: Hardware setup, Resources, Purwowibowo: Hardware setup, Data acquisition, Jalu Ahmad Prakosa: Writing-review, Data acquisition.

Funding statement. The authors received a research grant from Kedeputian IPT-LIPI (2020-2021) for supporting this work. However, we received no funding for the publication of this article.

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] N. R. Jones, Z. U. Qureshi, R. J. Temple, J. P. J. Larwood, T. Greenhalgh, and L. Bourouiba, "Two metres or one: what is the evidence for physical distancing in covid-19?," *BMJ*, vol. 370, p. m3223, Aug. 2020, doi: 10.1136/bmj.m3223.

[2] D. Bergman, C. Bethell, N. Gombojav, S. Hassink, and K. C. Stange, "Physical Distancing With Social Connectedness," *Ann. Fam. Med.*, vol. 18, no. 3, pp. 272–277, May 2020, doi: 10.1370/afm.2538.

[3] M. L. Parra Gordo, G. Buitrago Weiland, M. Grau García, and G. Arenaza Choperena, "Radiologic aspects of COVID-19 pneumonia: Outcomes and thoracic complications," *Radiol. (English Ed.*, vol. 63, no. 1, pp. 74–88, Jan. 2021, doi: 10.1016/j.rxeng.2020.11.002.

[4] N. B. Masters *et al.*, "Social distancing in response to the novel coronavirus (COVID-19) in the United States," *PLoS One*, vol. 15, no. 9, pp. 1–12, Sep. 2020, doi: 10.1371/journal.pone.0239025.

[5] I. Kuitunen, M. Artama, L. Mäkelä, K. Backman, T. Heiskanen-Kosma, and M. Renko, "Effect of Social Distancing Due to the COVID-19 Pandemic on the Incidence of Viral Respiratory Tract Infections in Children in Finland During Early 2020," *Pediatr. Infect. Dis. J.*, vol. 39, no. 12, pp. e423–e427, Dec. 2020, doi: 10.1097/INF.0000000000002845.

[6] C. T. Nguyen *et al.*, "A Comprehensive Survey of Enabling and Emerging Technologies for Social Distancing—Part I: Fundamentals and Enabling Technologies," *IEEE Access*, vol. 8, pp. 153479–153507, 2020, doi: 10.1109/ACCESS.2020.3018140.

[7] C. T. Nguyen *et al.*, "A Comprehensive Survey of Enabling and Emerging Technologies for Social Distancing—Part II: Emerging Technologies and Open Issues," *IEEE Access*, vol. 8, pp. 154209–154236, 2020, doi: 10.1109/ACCESS.2020.3018124.

[8] S. Saponara, A. Elhanashi, and A. Gagliardi, "Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19," *J. Real-Time Image Process.*, vol. 18, no. 6, pp. 1937–1947, Dec. 2021, doi: 10.1007/s11554-021-01070-6.

[9] S. Suryadi, E. Kurniawan, H. Adinanta, B. H. Sirenden, J. A. Prakosa, and P. Purwowibowo, "On the Comparison of Social Distancing Violation Detectors with Graphical Processing Unit Support," in *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 2020, pp. 337–342, doi: 10.1109/ICRAMET51080.2020.9298574.

[10] M. Rezaei and M. Azarmi, "DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic," *Appl. Sci.*, vol. 10, no. 21, pp. 1–29, Oct. 2020, doi: 10.3390/app10217514.

[11] A. Rahim, A. Maqbool, and T. Rana, "Monitoring social distancing under various low light conditions with deep learning and a single motionless time of flight camera," *PLoS One*, vol. 16, no. 2, pp. 1–19, Feb. 2021, doi: 10.1371/journal.pone.0247440.

[12] E. Jeong, J. Kim, and S. Ha, "TensorRT-based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards," *ACM Trans. Embed. Comput. Syst.*, pp. 1–26, Jan. 2022, doi: 10.1145/3508391.

[13] D.-J. Shin and J.-J. Kim, "A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform," *Appl. Sci.*, vol. 12, no. 8, pp. 1–19, Apr. 2022, doi: 10.3390/app12083734.

[14] S. Mittal, "A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform," *J. Syst. Archit.*, vol. 97, no. January, pp. 428–442, Aug. 2019, doi: 10.1016/j.sysarc.2019.01.011.

[15] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep Learning on Mobile and Embedded Devices," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–37, Jul. 2021, doi: 10.1145/3398209.

[16] A. Osipov *et al.*, "Identification and Classification of Mechanical Damage During Continuous Harvesting of Root Crops Using Computer Vision Methods," *IEEE Access*, vol. 10, pp. 28885–28894, 2022, doi: 10.1109/ACCESS.2022.3157619.

[17] Z. Lingxin, S. Junkai, and Z. Baijie, "A review of the research and application of deep learning-based computer vision in structural damage detection," *Earthq. Eng. Eng. Vib.*, vol. 21, no. 1, pp. 1–21, Jan. 2022, doi: 10.1007/s11803-022-2074-7.

[18] B. Kim, N. Yuvaraj, H. W. Park, K. R. S. Preethaa, R. A. Pandian, and D.-E. Lee, "Investigation of steel frame damage based on computer vision and deep learning," *Autom. Constr.*, vol. 132, no. September, p. 103941, Dec. 2021, doi: 10.1016/j.autcon.2021.103941.

[19] D. Wang, "Intelligent Detection of Vehicle Driving Safety Based on Deep Learning," *Wirel. Commun. Mob. Comput.*, vol. 2022, pp. 1–11, Jun. 2022, doi: 10.1155/2022/1095524.

[20] J. Ni, Y. Chen, Y. Chen, J. Zhu, D. Ali, and W. Cao, "A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods," *Appl. Sci.*, vol. 10, no. 8, pp. 1–29, Apr. 2020, doi: 10.3390/app10082749.

[21] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, "Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4316–4336, Jul. 2021, doi: 10.1109/TITS.2020.3032227.

[22] R. Ayachi, M. Afif, Y. Said, and M. Atri, "Traffic Signs Detection for Real-World Application of an Advanced Driving Assisting System Using Deep Learning," *Neural Process. Lett.*, vol. 51, no. 1, pp. 837–851, Feb. 2020, doi: 10.1007/s11063-019-10115-8.

[23] Z. Wu, X. Wang, and C. Chen, "Research on Lightweight Infrared Pedestrian Detection Model Algorithm for Embedded Platform," *Secur. Commun. Networks*, vol. 2021, pp. 1–7, Nov. 2021, doi: 10.1155/2021/1549772.

[24] H. Wu, Y. Hua, H. Zou, and G. Ke, "A lightweight network for vehicle detection based on embedded system," *J. Supercomput.*, pp. 1-16., Jun. 2022, doi: 10.1007/s11227-022-04596-z.

[25] A. K. A. Al Ghanadi, W. Mateen, and R. G. Ramaswamy, *Ilias Maglogiannis Lazaros Iliadis Applications*. Springer, Cham, 2020. Available at: Google Scholar.

[26] H. Adinanta, E. Kurniawan, Suryadi, and J. A. Prakosa, "Physical Distancing Monitoring with Background Subtraction Methods," in *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 2020, pp. 45–50, doi: 10.1109/ICRAMET51080.2020.9298687.

[27] S. Lu, B. Wang, H. Wang, L. Chen, M. Linjian, and X. Zhang, "A real-time object detection algorithm for video," *Comput. Electr. Eng.*, vol. 77, pp. 398–408, Jul. 2019, doi: 10.1016/j.compeleceng.2019.05.009.

[28] L. Jiao *et al.*, "A Survey of Deep Learning-Based Object Detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[29] L. Zhao and S. Li, "Object Detection Algorithm Based on Improved YOLOv3," *Electronics*, vol. 9, no. 3, pp. 1–11, Mar. 2020, doi: 10.3390/electronics9030537.

[30] A. M. Roy, R. Bose, and J. Bhaduri, "A fast accurate fine-grain object detection model based on YOLOv4 deep neural network," *Neural Comput. Appl.*, vol. 34, no. 5, pp. 3895–3921, Mar. 2022, doi: 10.1007/s00521-021-06651-x.

[31] F. Abdurahman, K. A. Fante, and M. Aliy, "Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models," *BMC Bioinformatics*, vol. 22, no. 1, pp. 1–17, Dec. 2021, doi: 10.1186/s12859-021-04036-4.

[32] C.-Y. Wang, H.-Y. Mark 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," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, vol. 2020-June, pp. 1571–1580, doi: 10.1109/CVPRW50498.2020.00203.

[33] NVIDIA, "NVIDIA TensorRT Documentation," *Nvidia Accelerated Computing*, 2022. [Online]. Available: docs.nvidia.com. [Accessed: 20-Jun-2022].

[34] A. Harvey and J. LaPlace, "Researchers Gone Wild," in *Practicing Sovereignty*, transcript Verlag, 2021, pp. 289–310. doi: 10.1515/9783839457603-016