An Investigation of YOLO-Based Helmet Detection for Workers on GPU and CPU Platforms

Kittakorn Viriyasatr[1] Warakorn Luangluewut[1, 2] Phunsak Thiennviboon[2*]
Achitpon Charoensuk[3] Piyarose Maleecharoen[1] Siraphob Santironnarong[1]
Ubon Thongsatapornwatana[1] and Pantape Kaewmongkol[1]

Received 29 April 2025, Revised 30 May 2025, Accepted 4 June 2025

Abstract

This research focused on enhancing workplace safety by detecting workers' helmet usage through the integration of artificial intelligence with closed-circuit television systems (CCTV). The detection process was based on object detection techniques using the YOLO (You Only Look Once) model family. This study evaluates the performance of the latest versions—YOLOv10 and YOLOv11—on both CPU and GPU platforms, considering both detection accuracy and inference speed. Among the tested model variants, a maximum processing speed of 454.52 frames per second was achieved on GPU and 34.01 frames per second on CPU, demonstrating their suitability for real-time CCTV-based applications. In terms of accuracy, YOLOv11n achieved a mean Average Precision (mAP@50) of 98.3% for helmet detection and 96.6% for head detection. The model also demonstrated strong classification performance, with precision and recall of 85.76% and 96.98% for the head class, and 93.02% and 96.56% for the helmet class, respectively. These results confirmed the model's effectiveness and balanced performance in distinguishing between helmeted and non-helmeted workers, supporting its potential for practical and cost-effective deployment in real-time safety monitoring systems.

Keywords: Closed Circuit Television System, YOLO, Helmet detection, CPU, GPU, Frame per second

[1] Research & Development Department, Defence Technology Institute
[2] Department of Electrical Engineering, Faculty of Engineering, Kasetsart University
[3] Faculty of Engineering King Mongkut's University of Technology North Bangkok
[*] Corresponding author, E-mail: phunsak.t@ku.th

การศึกษาการตรวจจับหมวกนิรภัยของผู้ปฏิบัติงาน โดยใช้ YOLO บนแพลตฟอร์ม GPU และ CPU

กิตตากร วิริยะศาสตร์[1] วรากร เลื่องลือวุฒิ[1, 2] พันศักดิ์ เทียนวิบูลย์[2*]
อชิตพล เจริญสุข[3] ปิยะรส มาลีเจริญ[1] สิรภพ สันติรณนรงค์[1]
อุบล ธงสถาพรวัฒนา[1] และ พันธุ์เทพ แก้วมงคล[1]

วันที่รับ 29 เมษายน 2568 วันที่แก้ไข 30 พฤษภาคม 2568 วันตอบรับ 4 มิถุนายน 2568

บทคัดย่อ

งานวิจัยนี้มุ่งเน้นการเพิ่มความปลอดภัยในสถานที่ทำงานโดยการตรวจจับการสวมหมวกนิรภัยของผู้ปฏิบัติงานผ่านการบูรณาการปัญญาประดิษฐ์เข้ากับระบบโทรทัศน์วงจรปิด ผ่านกระบวนการตรวจจับวัตถุด้วยชุดแบบจำลองตระกูล YOLO (You Only Look Once) โดยงานวิจัยนี้ได้ประเมินประสิทธิภาพของแบบจำลองเวอร์ชันล่าสุด ได้แก่ YOLOv10 และ YOLOv11 บนแพลตฟอร์มทั้ง CPU และ GPU โดยพิจารณาทั้งในด้านความแม่นยำในการตรวจจับและความเร็วในการประมวลผล จากแบบจำลองที่ทดสอบ พบว่า แบบจำลองสามารถประมวลผลได้สูงสุดถึง 454.52 เฟรมต่อวินาที บน GPU และ 34.01 เฟรมต่อวินาที บน CPU ซึ่งแสดงให้เห็นถึงความเหมาะสมในการใช้งานร่วมกับระบบโทรทัศน์วงจรปิดแบบเรียลไทม์ ในด้านความแม่นยำ YOLOv11n ให้ค่าความแม่นยำเฉลี่ย (mAP@50) เท่ากับ 98.3% สำหรับการตรวจจับหมวกนิรภัย และ 96.6% สำหรับการตรวจจับศีรษะ รวมถึงแสดงให้เห็นถึงประสิทธิภาพในการจำแนกประเภท โดยมีค่าความแม่นยำ (Precision) และค่าการเรียกคืน (Recall) เท่ากับ 85.76% และ 96.98% สำหรับคลาสศีรษะ และ 93.02% และ 96.56% สำหรับคลาสหมวกนิรภัย ตามลำดับ ผลการทดลองยืนยันถึงประสิทธิภาพและความสมดุลของแบบจำลองในการจำแนกระหว่างผู้ที่สวมและไม่สวมหมวกนิรภัย ซึ่งสนับสนุนศักยภาพของแบบจำลองนี้ในการนำไปประยุกต์ใช้จริงกับระบบตรวจสอบความปลอดภัยที่มีต้นทุนต่ำและมีประสิทธิภาพ

คำสำคัญ: ระบบโทรทัศน์วงจรปิด (CCTV), YOLO, การตรวจจับหมวกนิรภัย, หน่วยประมวลผลกลาง, หน่วยประมวลผลกราฟิก, จำนวนเฟรมต่อวินาที

---

[1] ฝ่ายวิจัยและพัฒนา, สถาบันเทคโนโลยีป้องกันประเทศ

[2] ภาควิชาวิศวกรรมไฟฟ้า, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเกษตรศาสตร์

[3] คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

[*] ผู้แต่ง, อีเมล: phunsak.t@ku.th

I. INTRODUCTION

In modern times, workplace safety in industrial environments has become increasingly critical due to the high incidence of accidents involving workers over the years [1] - [2]. One key safety measure is requiring workers to wear Personal Protective Equipment (PPE) to ensure their safety in any environment where workers are exposed to hazards that can cause injury or illness such as construction sites. Recently, advancements in computer vision technology [3] have been applied to worker safety in various scenarios [4] - [7], particularly for detecting whether workers are properly equipped with the necessary safety gear in designated work areas. This study specifically explores the use of the latest YOLO model versions for such safety-related tasks. YOLO models are well known for their superior speed and efficiency compared to other object-detection algorithms [8], making them particularly suitable for real-time safety monitoring systems.

While previous studies have applied YOLO to safety-related tasks [5], this study focused on evaluating the performance of the latest YOLO models—YOLOv10 and YOLOv11 (in both nano and extra-large variants)—for potential use in workplace safety applications.Instead of implementing a physical system, we conceptually adopted a CCTV-based configuration as described in [9] - [10] and assessed the models on both CPU and GPU platforms to examine whether their accuracy and frame rate were suitable for practical deployment. The goal was to determine whether CPU-based inference, which offered a significantly lower cost than GPU-based systems, could still deliver acceptable performance in real-time safety monitoring. If acceptable speed and accuracy can be maintained without GPUs [4] - [7], this could enable more cost-effective system design, particularly in selecting appropriate servers and hardware for specific environments.

II. BACKGROUND

1. YOLO (You Only Look Once)

YOLO, first introduced by J. Redmon et al. [11], is a family of single-stage deep learning models for object detection. Unlike two-stage detectors that first generate region proposals and then classify them, YOLO performs all tasks in a single forward pass. It predicts the bounding box coordinates, an objectness score (indicating the likelihood that an object is present), and the class probabilities for each detected object. This approach treats object detection as a single regression problem. Early versions of YOLO used a fixed grid structure (7×7 in YOLOv1), where each grid cell was responsible for predicting a few boxes. Later versions improved performance by introducing anchor boxes, multi-scale features, and anchor-free detection heads. Most versions used a technique called Non-Maximum Suppression (NMS) to remove overlapping boxes, but the newest versions—such as YOLOv10 and YOLOv11—replace NMS with end-to-end strategies that still follow the same "look once" principle.

YOLO is known for three main strengths: real-time speed, simple and unified architecture, and strong accuracy. A recent benchmark comparing YOLOv1 through YOLOv11 shows steady improvements in detection accuracy (mAP), processing speed, and efficiency. These developments are clearly illustrated in the Figure 1. titled Evolution of YOLO model versions, adapted from N. Jegham et al. [12]. Because of its speed, open-source availability, and ease of use, YOLO is widely used in both research and real-world applications that require fast and accurate object detection. This work focused on the most recent versions—YOLOv10 and YOLOv11—in both nano (-n) and extra-large (-x) variants, offering a practical comparison between lightweight and high-performance models. YOLOv10 and YOLOv11 are the latest versions in the YOLO family, incorporating architectural improvements that further enhance detection performance, particularly in terms of speed and efficiency.

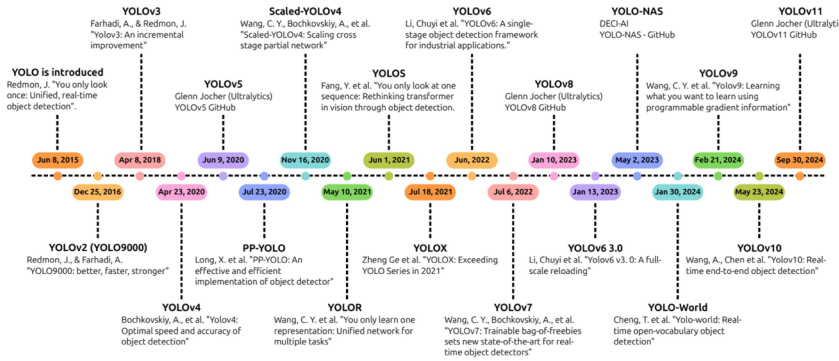

Fig.1 Evolution of YOLO model versions, adapted from N. Jegham *et al.* [12].

2. Model Evaluation

This study evaluated the model's performance using four metrics derived from the confusion matrix [13] - [14]. Let *TP, FP,* and *FN* denote the number of true positive (correct positive prediction), false positive (incorrect positive prediction), and false negative (incorrect negative prediction) samples, respectively. The definitions of these metrics are as follows:

- **Precision** measures the accuracy of positive predictions (ranging from 0 to 1) and is calculated with Equation (1).

$$\text{Precision} = \frac{TP}{TP+FP} \quad , \tag{1}$$

- **Recall** measures the accuracy of predicting actual positive samples (ranging from 0 to 1) and is calculated with Equation (2).

$$\text{Recall} = \frac{TP}{TP+FN} \quad , \tag{2}$$

- **Average Precision (AP)** measures the area under the precision-recall (PR) curve (ranging from 0 to 1) and is calculated using Equation (3).

$$AP = \int_0^1 p(r)dr \, , \tag{3}$$

where p and r are precision and recall, respectively.

- **Mean Average Precision (mAP)** measures the average of the Average Precision (AP) scores across all classes. **mAP@50** refers to mAP evaluated at a 50% overlap threshold between predicted and ground truth bounding boxes, and it is a commonly used metric for assessing the accuracy of object detection algorithms.

3. Benchmark Frame Rate in CCTV Systems

This section discusses the relevance of processing speed in CCTV systems, referencing prior works that have proposed cost-effective solutions using Raspberry Pi devices [9] - [10]. In particular, we focused on the system developed by A. Kurniawan *et al.* [10], which employed a Raspberry Pi as a server and is conceptually illustrated in Figure 2. This Raspberry Pi-based CCTV system achieved frame rates ranging from 10 to 30 frames per second (fps), depending on the resolution. We later referred to this frame rate range as a practical benchmark for real-time or near-real-time CCTV applications in constrained environments.

However, when considering AI-powered applications, the Raspberry Pi may not be suitable due to its limited computational power, restricted memory, and lack of dedicated AI accelerators such as GPUs or TPUs, which are essential for efficiently handling complex models and high-speed data processing [15]. Therefore, in this study, we explored the feasibility of using the same CCTV system while replacing the Raspberry Pi-based server with CPU- and GPU-based servers for object detection tasks. Our objective was to evaluate the performance and processing speed (in fps)

of various YOLO algorithms on these alternative computing platforms, using the Raspberry Pi's frame rate range as a reference constraint.

Fig. 2 Conceptual illustration of a CCTV camera system, adapted from A. Kurniawan *et al.* [10].

4. Model scaling

Model scaling [8] refers to the process of adjusting the size of a deep learning model by modifying parameters such as **depth** and **width**, in order to tailor the model to specific resource constraints and performance requirements. In the case of YOLO, model scaling primarily involves two parameters.

- **Depth Multiple** adjusts the number of layers in the model to increase or decrease its depth.
- **Width Multiple** adjusts the number of channels in each layer to increase or decrease its width.

By tuning these parameters, YOLO can be scaled to create models of different sizes according to task requirements. For instance, smaller models may offer faster computational speeds, while larger models can provide improved accuracy. The YOLO family includes several model variants, each with different scaling configurations.

III. METHODOLOGY

1. Data Preparation

The dataset used in this study is the publicly available Hard Hat Workers Dataset [16], which includes annotated images of people with and without helmets. The annotations were prepared using two class labels: class 0 for heads without helmets, and class 1 for heads with helmets. Example images from the dataset are shown in Figure 3.

The training and testing processes were conducted using Google Colab [17], which provides access to a variety of CPU and GPU configurations as shown in Figure 4. This study aims to assess the performance of different hardware types—both in terms of detection accuracy and inference speed (frame rate)—to determine whether cost-effective CPU-based deployment is feasible for real-time safety monitoring.

Fig. 3 Sample images used for data preparation, including individuals with and without helmets. To protect personal privacy, certain details in the images have been censored.

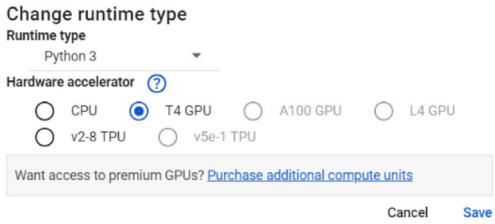

Fig. 4 Hardware accelerators available on Google Colab, including various CPU and GPU configurations used in this study.

The tested GPU models included NVIDIA A100 and NVIDIA T4 [18], while the CPU models included an Intel Xeon processor (CPU-Xeon) [19] and an AMD EPYC processor (CPU-AMD) [20]. These setups were evaluated to simulate their use as potential server-side processing units for a CCTV-based safety system running the YOLO model.

2. Model Training

After data preparation, a subset of the labeled dataset was used to train object detection models. In total, 4,858 images were used for training and 1,464 images for validation. The validation set contained 1,505 labeled instances of people without helmets and 4,021 labeled instances of people wearing helmets. This study evaluated four YOLO models to compare their performance in terms of speed and accuracy. The selected models included YOLOv10n, YOLOv10x, YOLOv11n, and YOLOv11x. These variants differed in model scale, specifically in their depth and width multipliers, which controlled the architecture's depth and width, respectively. All images used for training had a resolution of 640×640 pixels. The Nano models (YOLOv10n and YOLOv11n) were designed for low-resource environments, offering faster inference and smaller model size, but typically with lower accuracy. In contrast, the Extra-Large models (YOLOv10x and YOLOv11x) offered higher detection accuracy but required more computational resources and run at slower speeds. All training was performed using the Ultralytics YOLO implementation in Python [21].

3. Model Testing

After training the four models—YOLOv10n, YOLOv10x, YOLOv11n, and YOLOv11x—we evaluated their performance using a separate test dataset to ensure no overlap with the training or validation data [12]. This helped to maintain the integrity of the evaluation and avoids overfitting. The test set consisted of 1,812 images, containing a total of 6,840 labeled instances, including 1,788 examples of people not wearing helmets (class 0) and 5,052 examples of people wearing helmets (class 1).

The goal of testing was to assess each model's accuracy and inference speed (frame rate) in object detection. All test images had a resolution of 640 × 640 pixels, consistent with the training process.

IV. RESULTS AND DISCUSSION

1. Training Results

All four models—YOLOv10n, YOLOv10x, YOLOv11n, and YOLOv11x—were trained for 100 epochs. The original training dataset was split into two mutually exclusive subsets for model training and validation during the training process. The results showed that all models achieved comparable performance, with mAP@50 stabilizing at approximately 97–98%. Examples of the training performance of the YOLOv11n model are illustrated in Figure 5.

During the validation phase, all YOLO models demonstrated high detection accuracy. In many cases, the confidence scores of correctly detected objects ranged from 80% to 90%. Representative validation results are shown in Figure 6.

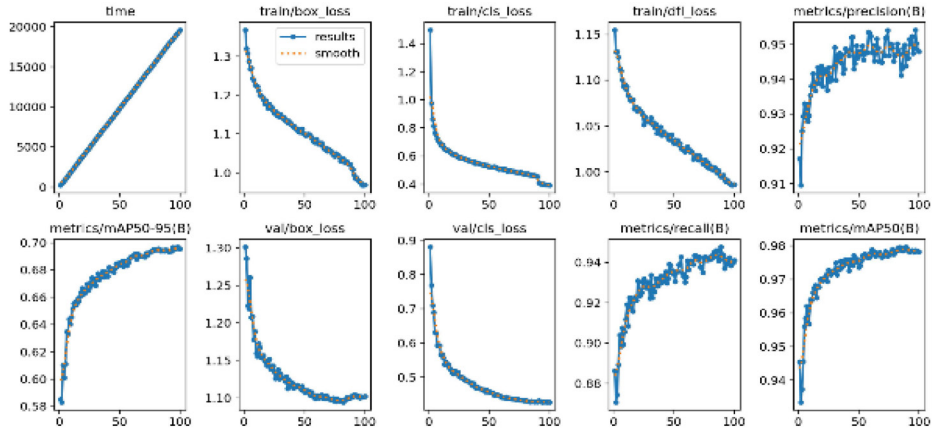

Fig. 5 Example training performance of the YOLOv11n model over 100 epochs, including loss, mAP@50, and other evaluation metrics.

Fig. 6 Example output from the YOLOv11n model detecting individuals wearing and not wearing helmets. *To protect personal privacy, certain details in the image have been censored.*

2. Test Results

The testing results for both YOLOv10 and YOLOv11 models are summarized in Tables I and II. The mAP@50 values are very similar across all variants, with approximately 96-97% for head detection and 98-99% for helmet detection. This also indicates that model scaling (Nano vs. Extra-Large) has little impact on detection accuracy.

In contrast, inference speed differs significantly depending on the model size. As expected, the Nano variants (YOLOv10n and YOLOv11n) run substantially faster than their Extra-Large counterparts across both CPU and GPU platforms. Notably, when running on CPUs, the Nano models achieved 21-34 frames per second (FPS)—a frame rate sufficient for many CCTV camera applications, such as the example described in Section II-3. These results suggest that CPU-based deployment can be a viable and cost-effective solution for real-time safety monitoring, especially in environments with limited computational resources. Using GPUs in such cases may represent an unnecessary use of resources.

Further insight is provided by the confusion matrix for the YOLOv11n model, shown in Figure 7. The model correctly detected 1,734 out of 1,788 head instances and 4,878 out of 5,052 helmet instances. This corresponds to a precision of 85.76% (head) and 93.02% (helmet), a recall of 96.98% (head) and 96.56% (helmet), and F1-scores of 91.02% (head) and 94.76% (helmet), respectively. Interestingly, misclassifications between head and helmet objects were relatively low, while most errors were due to missed detections or false positives involving the background.

V. CONCLUSION

YOLOv11 demonstrated slightly higher accuracy than YOLOv10 across all test scenarios. For head detection, YOLOv11 achieved mAP@50 values between 0.966 and 0.968, compared to 0.961 for YOLOv10. For helmet detection, both models performed similarly well, with mAP@50 values ranging from 0.981 to 0.986, indicating consistently strong detection capabilities.

TABLE I. Testing Results for YOLOv10n and YOLOv10x models (Nano and Extra-Large variants) across Various Hardware Platforms

Model	Hardware accelerator	mAP@50 (Head)	mAP@50 (Helmet)	Speed (FPS)
YOLOv10n	GPU-A100	0.961	0.981	454.52
	GPU-L4			416.71
	GPU-T4			169.51
	CPU-Xeon			30.96
	CPU-AMD			21.55
YOLOv10x	GPU-A100	0.961	0.982	185.20
	GPU-L4			46.95
	GPU-T4			25.58
	CPU-Xeon			2.256
	CPU-AMD			1.447

TABLE II. Testing Results for YOLOv11N and YOLOv11X Models (Nano and Extra-Large variants) across Various Hardware Platforms

Model	Hardware accelerator	mAP@50 (Head)	mAP@50 (Helmet)	Speed (FPS)
YOLOv11n	GPU-A100	0.966	0.983	384.62
	GPU-L4			344.83
	GPU-T4			158.73
	CPU-Xeon			34.01
	CPU-AMD			25.77
YOLOv11x	GPU-A100	0.968	0.986	169.49
	GPU-L4			44.84
	GPU-T4			21.28
	CPU-Xeon			2.39
	CPU-AMD			1.31

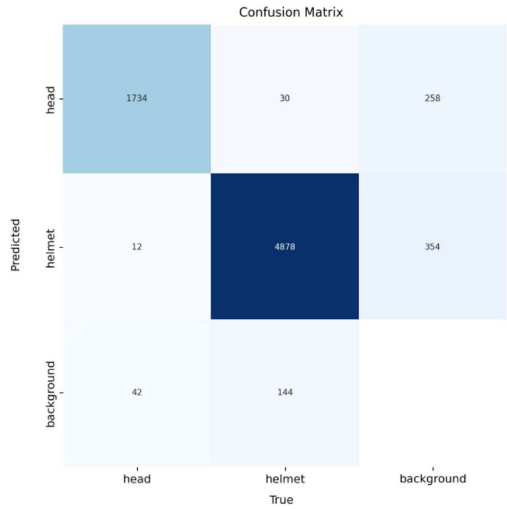

Fig. 7 Confusion matrix for the YOLOv11n model evaluated on the test dataset. Ground-truth classes include only head and helmet. The "background" row represents missed detections (i.e., true head or helmet objects that were not detected), while the "background" column represents false positives—predicted objects that do not correspond to any labeled object. Precision and recall are reported only for the head and helmet classes, as background is not an annotated class.

In terms of inference speed, the GPU-A100 delivered the highest performance, with YOLOv10n reaching 454.52 FPS and YOLOv11n achieving 384.62 FPS. Other GPUs, such as the T4 and L4, provided lower but still acceptable frame rates for many real-time applications. In contrast, CPUs—including Intel Xeon and AMD EPYC—produced significantly slower speeds, with Xeon achieving 31–34 FPS and AMD 21–26 FPS. Despite the speed advantage of GPUs, CPU-based deployment remains a cost-effective and practical alternative—especially for CCTV monitoring systems. Notably, Nano-scaled YOLO models on CPUs deliver detection accuracy comparable to their GPU-based counterparts, making them suitable for resource-constrained environments where cost-efficiency is prioritized over maximum processing speed.

VI. ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the Research & Development Department, Defence Technology Institute (DTI), Nonthaburi, Thailand, for their invaluable support and resources, which played a vital role in the success of this research. Special thanks are extended to the Director and members of the research division at DTI for their guidance and encouragement throughout the project. The authors also gratefully acknowledge the Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand, for their technical assistance, expertise, and collaboration, which were instrumental in achieving the objectives of this study. Finally, the authors thank all individuals who contributed to this research, including those not mentioned by name, for their support and encouragement throughout the project.

VII. REFERENCES

[1] CBC News. "Surveyor Killed in Quebec Road Construction Accident." CBC.ca. http://www.cbc.ca/news/canada/story/2011/09/02/worker-death-chateauguay.html (accessed Aug. 3, 2013).

[2] CBC News. "3 Most Dangerous Job Sectors in Canada." CBC.ca. http://www.cbc.ca/news/canada/story/2012/04/25/f-dangerous-jobs.html (accessed Aug. 3, 2013).

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Comput. Intell. Neurosci.*, vol. 2018, no. 1, 2018, doi: 10.1155/2018/7068349.

[4] M. - W. Park, N. Elsafty, and Z. Zhu, "Hardhat-wearing detection for enhancing on-site safety of construction workers," *J. Constr. Eng. Manag.*, vol. 141, no. 9, p. 04015024, 2015, doi: 10.1061/(ASCE)CO.1943-7862.0000974.

[5] E. Wanlin, Z. Yang, and J. Yu, "Detection and Recognition of Personal Protection Equipment Wearing Based on an Improved YOLOv5 Algorithm,"

in *2023 4th Int. Symp. Comput. Eng. Intell. Commun. (ISCEIC),* Nanjing, China, 2023, pp. 229 - 232, doi: 10.1109/ISCEIC59030.2023.10271227.

[6] S. Chen, W. Wang, Y. Ouyang, H. Zhu, T. Ji, and W. Tang, "Detection of Safety Helmet Wearing Based on Improved Faster R-CNN," in *2020 Int. Joint Conf. Neural Netw. (IJCNN),* Glasgow, UK, Jul. 2020, pp. 1 - 7, doi: 10.1109/IJCNN48605.2020.9207574.

[7] J. Hu, X. Gao, H. Wu, and S. Gao, "Detection of Workers Without Helmets in Videos Based on YOLO V3," in *2019 12th Int. Congr. Image Signal Process., Biomed. Eng. Inform. (CISP-BMEI),* Oct. 2019, pp. 1 - 4, doi: 10.1109/CISP-BMEI48845.2019.8966045.

[8] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *2020 IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR),* Seattle, WA, USA, 2020, pp. 10781 - 10790, doi: 10.1109/CVPR42600.2020.01079.

[9] E. Rohadi *et al.*, "Internet of Things: CCTV Monitoring by Using Raspberry Pi," in *2018 Int. Conf. Appl. Sci. Technol. (iCAST),* Oct. 2018, pp. 454 - 457, doi: 10.1109/iCAST1.2018.8751612.

[10] A. Kurniawan, A. Ramadlan, and E. M. Yuniarno, "Speed Monitoring for Multiple Vehicle Using Closed Circuit Television (CCTV) Camera," in *2018 Int. Conf. Comput. Eng. Netw. Intell. Multimedia (CENIM),* Surabaya, Indonesia, Nov. 2018, pp. 88 - 93, doi: 10.1109/CENIM.2018.8710854.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conf. Comput. Vision Pattern Recognit. (CVPR),* Jun. 2016, pp. 779 - 788, doi: 10.1109/CVPR.2016.91.

[12] N. Jegham, C. Y. Koh, M. F. Abdelatti, and A. Hendawi, "Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors," 2024, arXiv:2411.00201.

[13] S. Visa, B. Ramsay, A. Ralescu, and E. van der Knaap, "Confusion Matrix-based Feature Selection," *CEUR Workshop Proc.*, vol. 710, pp. 120 - 127, 2011.

[14] A. Subasi, *Practical Machine Learning for Data Analysis Using Python.* Academic Press, 2020.

[15] P. Steadman, P. Jenkins, R. S. Rathore, and C. Hewage, "Challenges in Implementing Artificial Intelligence on the Raspberry Pi 4, 5 and 5 with AI HAT," in *Int. Conf. Comput. Commun., Cybersecurity AI,* London, UK, 2024, pp. 147 - 157.

[16] Roboflow. "Hard Hat Workers Dataset." PUBLIC. ROBOFLOW.ai. https://public.roboflow.ai/object-detection/hard-hat-workers (accessed: Dec. 24, 2024).

[17] E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform,* 1st ed. Berkeley, CA, USA: Apress, 2019, pp. 59 - 64, doi: 10.1007/978-1-4842-4470-8_7.

[18] NVIDIA, "GPU Positioning for Virtualized Compute and Graphics Workloads," NVIDIA, Tech. Rep. TB-09867-001_v03, Accessed: Dec. 24, 2024. [Online]. Available: https://images.nvidia.com/data-center/technical-brief-gpu-positioning-virtualized-compute-and-graphics-workloads.pdf

[19] Intel, "Measuring Processor Power: TDP vs. ACP Whitepaper," Accessed: Dec. 24, 2024. [Online]. Available: https://www.intel.com/content/dam/doc/white-paper/resources-xeon-measuring-processor-power-paper.pdf

[20] AMD, "5th Gen AMD EPYC Processor Architecture Whitepaper," Accessed: Dec. 24, 2024. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/5th-gen-amd-epyc-processor-architecture-white-paper.pdf

[21] Ultralytics. "Ultralytics YOLO11." DOCS. ULTRALYTICS.com. https://docs.ultralytics.com/models/yolo11/ (accessed May 19, 2025).