International Journal of Safety and Security Engineering
Vol. 15, No. 5, May, 2025, pp. 1075-1088
Journal homepage: http://iieta.org/journals/ijsse

YOLO-Based Helmet Detection and Intelligent Parking System: A Case Study at Diponegoro University

Agustinus Adven Christo, M. Bintang Prayoga Utama, Yosia Aser Camme, Dania Eridani*, Patricia Evericho Mountaines

Department of Computer Engineering, Faculty of Engineering, Diponegoro University, Semarang 50275, Indonesia

Corresponding Author Email: dania@ce.undip.ac.id

https://doi.org/10.18280/ijsse.150520

Received: 16 April 2025
Revised: 17 May 2025
Accepted: 25 May 2025
Available online: 31 May 2025

Keywords:
helmet detection, intelligent parking system, embedded system, YOLO, deep learning, raspberry Pi 5, Arduino uno, Hailo-8L

ABSTRACT

Ensuring helmet compliance in motorcycle parking areas remains a challenge due to the ineffectiveness of the manual checking system. While substantial research has been undertaken on smart parking systems, their integration with driving safety compliance remains limited. This work presents a novel intelligent barrier gate control system that uses deep learning-based object detection to automate helmet compliance verification. The system uniquely combines several advanced technologies. An Arduino Uno operates the physical barrier gate, while a Raspberry Pi 5 processes real-time video input to detect helmet usage using a YOLO based deep learning model. Hailo-8L AI accelerator is utilized to enhance efficiency. A custom dataset of 3,146 labeled images (2,202 for training, 630 for samples, and 314 for testing) was used to train the model. Performance evaluation shows that the system achieves a mean Average Precision (mAP) of 0.9825, indicating high detection accuracy. Real-world testing in the Diponegoro University campus parking lot demonstrates consistent barrier gate operation, with response times ranging from 6 to 7.8 seconds and an average of 7.11 seconds. The system also shows optimized low power consumption of only 2.3W. This system addresses manual checks at Diponegoro University and supports automated safety enforcement and intelligent access control.

1. INTRODUCTION

Many segments of Indonesian society, especially university students, now consider motorcycles an essential transportation requirement. Badan Pusat Statistik (BPS) statistics show Indonesia recorded 125.305.332 motorcycles in 2022. Based on the same statistic, there were 139.258 road accidents in Indonesia, a considerable rise of 35.613 instances over the year before [1]. This indicates that the large number of motorcycle riders does not translate into a commensurate level of compliance with traffic laws, such as helmet usage [2].

Non-compliance with helmet use is one of the main factors contributing to the high number of fatal accidents, since helmets are crucial in protecting riders from fatal head injuries [3]. In Indonesia, the automated surveillance system for driving safety compliance has been implemented on highways to detect and take action against traffic violations through the implementation of Electronic Traffic Law Enforcement (ETLE) [4]. However, a similar system has yet to be implemented in campus areas that also experience high traffic density, such as Diponegoro University. At the campus, driving compliance inspections, including helmet use, are conducted manually by security staff at the entrances of each faculty parking area before riders are allowed to enter.

While helmet compliance enforcement is feasible, it poses obstacles in Diponegoro University parking spaces due to the lack of an automated barrier gate system. Without a system to control which riders are permitted entry, motorcycles can pass through unchecked, making it challenging to enforce helmet regulations effectively. The dependence on manual inspections by security personnel is inefficient and prone to human mistakes, particularly during busy periods when many vehicles enter simultaneously.

Automated parking systems have been extensively researched to improve efficiency and security in vehicle access control. A comprehensive review emphasizes various research on smart parking solutions, including methodologies, sensor technologies, networking strategies, and computational approaches. These systems generally use sensors, camera-based images, or a mix of the two to assess whether a parking space is occupied [5]. However, these studies are primarily concerned with parking space management to detect the availability of parking slots for motorcycle riders.

While considerable research has focused on parking availability, applying comparable technology to monitor compliance with safety regulations, such as helmet use, remains limited. With the growing number of two-wheeled vehicles and the related risk of accidents, a YOLO-based helmet detection system might provide a solution to improve road safety. Our work explores the development of a parking system that integrates helmet compliance monitoring with an automatic barrier gate within a parking area.

Machine learning with object detection is frequently used to automate driving safety compliance inspections, such as helmet detection. One popular approach is YOLO (You Only Look Once), which detects things in real time. YOLO is a Deep Learning Algorithm that can determine whether motorcycle riders are wearing helmets using a trained dataset [6].

Helmet detection for traffic violation monitoring has become a notable application of the YOLO algorithm. By leveraging YOLOv8, a system could detect motorcycles, riders, and helmets from real-time video footage. This method enables automatic recognition of helmet compliance without needing direct human involvement, making it an achievable option for increasing road safety enforcement. The model works by first identifying motorbikes and their riders inside a given frame, then performing a classification step to assess the existence of a helmet. By training on a diversified dataset, YOLOv8 improves detection accuracy over various environmental conditions, assuring reliability even in complex environments. The capacity to evaluate video data in real time makes this technique ideal for high-traffic locations where human monitoring may be ineffective [7].

Multiple research studies have demonstrated that automated barrier gate systems based on embedded devices such as Raspberry Pi and Arduino improve efficiency by replacing manual procedures with intelligent access control methods, thereby enhancing security and parking management [8-10]. These solutions enhance access control efficiency and provide a foundation for integrating machine learning-based compliance monitoring. A smart parking system might be developed by combining Raspberry Pi's computing capacity with YOLO for real-time helmet identification and Arduino for automated barrier gate control, allowing for the enforcement of helmet regulations while improving overall parking management on campus.

An AI accelerator such as the Hailo-8L can speed up real-time detection to increase the system's efficiency. The Hailo-8L AI processor is designed to run deep learning models quickly and efficiently [11]. This technique is especially well-suited for campus traffic monitoring systems because it allows local data processing without relying on significant computational resources. Hailo-8L enables the system to detect helmets with high accuracy and fast reaction times. This allows the automated barrier gate to make real-time decisions, such as denying entry to riders who violate helmet regulations.

Therefore, the design of a Barrier Gate Control Technology for Parking Systems that incorporates deep learning into automated barrier gates and an AI accelerator like Hailo-8L offers a novel integration of multiple subsystems into a unified, intelligent solution. Unlike previous researches that address helmet detection or access control separately, this research unifies real-time safety regulation enforcement with physical access control. This strategy enables direct processing on edge devices using YOLO-based helmet detection models optimized by Hailo-8L, ensuring fast and accurate inference directly. Combined with automated barrier gate control and a hardware-efficient setup using Raspberry Pi and Arduino, this solution offers a practical approach for campus-scale deployment. The novelty lies in the end-to-end integration of computer vision, edge AI acceleration, and physical gate control, enabling automatic access control in real time. This approach offers a scalable and practical solution for improving traffic safety, while accelerating the campus's transformation into a smarter and automated environment.

2. LITERATURE REVIEW

2.1 Hardware technologies in barrier gate control

Previous research has implemented various hardware components to automate barrier gate systems. The system in the study [8] shows how an Arduino-based controller may control a gate mechanism by utilizing an Arduino Mega, a DC motor, RFID, and ultrasonic sensors. Likewise, the study [9] offers an Arduino Uno-based smart parking system. Meanwhile, this research [10] presents a more advanced setup by integrating a Raspberry Pi 3 as the primary processing unit in an automated parking system. The system utilizes an ultrasonic sensor, a wired camera, and a DC motor to perform license plate recognition using the YOLO algorithm. The Raspberry Pi operates as a standalone computing device, responsible for processing image data from the camera, and controlling the stepper motor to open or close the barrier gate based on the recognized license plate, demonstrates that the combination of YOLO and Raspberry Pi can be effectively applied to improve the efficiency of automated barrier gate.

These implementations show that Arduino and Raspberry Pi are commonly used in automated barrier gates, with each component playing a distinct role, such as motor control, object detection, and real-time decision-making. These findings form a foundation for the proposed system.

This research utilizes various available microcontrollers and single-board computers, specifically the Arduino Uno and Raspberry Pi 5. The Arduino Uno is selected due to its simplicity, broad community support, and sufficient I/O capabilities [12] to control actuators and sensors in automated barrier gates. The Raspberry Pi 5, the latest generation in the Raspberry Pi family [13] offers significant processing power and memory bandwidth improvements, making it suitable for executing deep learning models such as YOLOv8. These choices balance performance and integration flexibility for the proposed system.

A linear actuator is a component that can extend and retract linearly. For centuries, such actuators have relied on hydraulic power [14]. Linear actuators play an essential role in creating accurate linear motion. These devices have been widely used in various applications, including automated vehicle steering systems, soft robotic actuators, prosthetics and rehabilitation, and automotive technologies [15]. Based on this research, we can use a linear actuator as a driving mechanism in an automated barrier gate.

A loop detector is an inductive sensor embedded in the ground that operates by detecting changes in the magnetic field caused by the presence of a vehicle. This sensor consists of a coil of wire installed in the ground and is connected to a control system capable of recognizing magnetic field disturbances when a vehicle passes or stops in a designated area [16]. According to the research, loop detectors are utilized in automated barrier gates to detect the presence of a vehicle.

2.2 Object detection in parking systems

Deep learning has emerged as a vital technique for generating accurate and efficient helmet detection systems. Unlike traditional rule-based methods, deep learning does not require manually constructed features, allowing models to generalize across various datasets and complicated urban traffic situations. Deep learning-based helmet detection systems, notably those based on CNNs and transformer

topologies, have been shown in research to improve detection accuracy and robustness considerably. Even when helmets are tiny, partially concealed, or visually similar to other headgear, the deep-learning approach detects helmets accurately and reliably [17]. These improvements have transformed deep learning into a vital tool for road safety enforcement, allowing for real-time, automated monitoring of helmet compliance to assist law enforcement and traffic management.

Convolutional Neural Networks (CNN) are a key deep learning approach in helmet identification, with architectures such as Faster R-CNN, SSD, and specifically YOLO capable of detecting helmet presence in various lighting conditions, angles, and backgrounds unds [18]. Deep learning models can automatically learn visual helmet characteristics through non-linear transformation layers without requiring complex manual feature engineering. Adapting to complex data allows helmet detection systems to operate with high accuracy, making them highly effective solutions for monitoring personal protective equipment compliance in industrial, construction, and transportation environments [17, 19].

Helmet detection for traffic violation monitoring has become a prominent application of the YOLO algorithm, allowing for automatic enforcement of helmet-wearing requirements among motorcycle riders. YOLOv8 enables a system to efficiently recognize motorcycles, riders, and helmets in real-time video footage, providing rapid and precise identification without human interaction. The detection process involves a two-step approach, where YOLOv8 first recognizes motorcycles and riders within a video frame and then classifies whether the detected rider is wearing a helmet. Training on a broad dataset increases the model's robustness, allowing it to adapt to changing lighting conditions, helmet designs, and angles. Compared to prior YOLO versions, YOLOv8 has higher accuracy, shorter inference, and better object tracking, making it especially effective in high-traffic areas where manual monitoring is unfeasible. Research has demonstrated that YOLOv8-based helmet detection systems reliably improve traffic safety enforcement and reduce accident risks. With its real-time processing abilities, this technique is suited for high-traffic areas where manual surveillance is impractical [7].

YOLOv8 has achieved strong performance metrics in motorcycle helmet detection tasks, confirming its suitability for real-time traffic monitoring. In one study, YOLOv8 attained a mean Average Precision (mAP@0.5:0.95) of 0.42675 for helmet detection, with class-specific mAP values of 0.914 for "with helmet" and 0.865 for "without helmet" categories [20]. In another investigation, YOLOv8 reached an overall mAP@0.5 of 0.85, with precision and recall values of 0.838 and 0.782 [21]. This level of accuracy ensures reliable identification of non-compliant riders even in complex urban settings characterized by high vehicle density and variable lighting. Compared to earlier YOLO versions, YOLOv8 not only reduces false detections but also accelerates inference, which is essential for real-time enforcement systems.

YOLOv8n, the nano variant of YOLOv8, is chosen in this system due to its suitability for edge processing in real-time environments. While newer versions may deliver slightly higher accuracy in general object detection tasks, research such as Vaikunth et al. [22] indicates that in specific applications like helmet detection, the performance gap between YOLOv8 and its successors is minimal or even reversed in specific scenarios. This trade-off makes YOLOv8n a practical and efficient choice for applications like helmet detection in proposed systems, where speed and hardware performance are crucial.

The impact of helmet detection technology goes beyond traffic surveillance and law enforcement, finding applications in access control and transportation safety systems. Helmet detection in parking systems can be used with automated access control devices, such as barrier gates, to ensure safety compliance. This technology improves parking facility security and regulatory compliance by permitting only fully equipped riders to gain entry. Combining YOLO-based object detection with an automated barrier gate significantly advances intelligent transportation systems, improving safety enforcement in real-world applications.

2.3 AI accelerator hardware

The Hailo-8 AI accelerator offers a significant leap in edge AI processing for object detection, providing more accuracy and less latency than traditional hardware such as the Jetson Orin Nano. The Hailo-8 improves accuracy by over 12% while reducing latency by 20 milliseconds, making it ideal for security and automation applications. This efficiency is due to its particular neural processing architecture, which allows for real-time inference with low power consumption. Unlike GPUs, which are frequently power-intensive and have higher processing times, the Hailo-8 is intended for low-power edge AI deployment, making it ideal for intelligent surveillance, transportation safety, and industrial automation [11].

The Hailo-8L, when paired with a Raspberry Pi 5, achieves low-latency inference (~53 ms) with higher frame rates (up to 42 FPS) and lower power consumption (10.4–10.6 W), compared to desktop GPU setups that require 45–50 W power while delivering lower frame rates (~16 FPS) and higher latency [23].

Previous research [24] showed that the use of the HEF (Hailo Executable Format) format optimized for the Hailo-8L accelerator significantly provides better performance than running the model using ONNX on the Raspberry Pi CPU. Tests show that processing using HEF with Hailo-8L can achieve speeds up to 15 times higher than execution on the Raspberry Pi CPU with ONNX, with a maximum frame rate of around 45 FPS versus only 3 FPS. In addition, the processing latency with HEF is also much lower, reducing the processing time from around 300 ms to less than 20 ms, which greatly supports real-time applications and rapid decision making. The use of HEF also shows efficiency in resource utilization and system stability in direct operations at the edge, without the need to rely on communication to the server or cloud.

YOLOv8 excels in real-time helmet detection, adapting to shifting conditions with remarkable speed and accuracy. Hailo-8, an AI accelerator, improves processing efficiency even further. Using YOLO-based detection, hardware selection, and Raspberry Pi processing results in a robust smart parking system that improves safety and comfort at Diponegoro University.

3. PROPOSED WORKS

3.1 System architecture

Figure 1 shows the proposed system architecture. It ensures automated helmet compliance enforcement while maintaining safe gate operation for motorcycle riders.

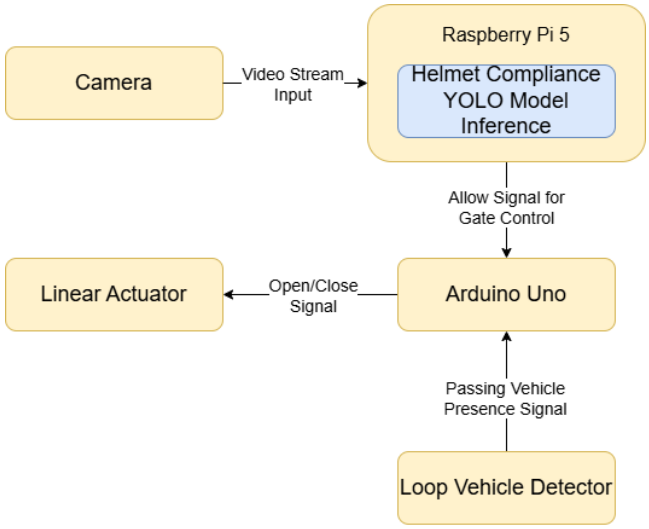


Figure 1. System architecture

The process starts with a camera capturing a video stream input sent to the Raspberry Pi 5. The Raspberry Pi 5 runs a YOLO-based helmet compliance model, which analyzes the video feed to identify whether or not the motorcycle rider is wearing a helmet. If compliance is identified, the Raspberry Pi 5 sends an accept signal to the Arduino Uno, which controls the gate. The gate is opened by the linear actuator, which is controlled by the Arduino Uno, upon receiving an acceptable signal. Meanwhile, a loop vehicle detector constantly detects vehicle presence and sends a signal to the Arduino Uno. This signal keeps the gate open when the motorcycle rider hasn't completely passed through it, preventing it from closing prematurely. Once the vehicle has completely passed and the loop detector no longer detects its presence, the Arduino Uno sends a close signal to the linear actuator, closing the gate. This method improves safety by allowing only helmet-wearing motorcycle riders to pass through and preventing the gate from shutting on a moving vehicle.

3.2 Datasets

This research acquired the dataset through primary data collection via manual photography using smartphone cameras. The entire data collection process was systematically conducted at the research location, specifically the parking area of the Department of Computer Engineering, Faculty of Engineering, Diponegoro University. A total of 3.146 sample images were collected, ensuring a diverse representation of various conditions in the parking area, including lighting variations (morning, noon, cloudy, and shaded conditions) and different camera angles (frontal, side, top-down) and distances (1-3 meters).

All image samples in this dataset were manually annotated using Roboflow. The annotation process involved identifying and labeling key objects in each image with bounding boxes and corresponding class labels. To ensure consistency and accuracy, every annotation was subsequently reviewed by a second annotator. A standardized labeling guideline was applied throughout the process: the "helmet" class includes various types of helmets worn by riders, excluding construction helmets, which are not part of the dataset. The "head" label is used to annotate individuals who are not wearing helmets, including those whose heads are covered by non-helmet items or accessories. The "motorcycle" class encompasses different types of two-wheeled motorcycles commonly used in the campus area. Lastly, the "person" class is reserved exclusively for pedestrians—motorcycle riders are not included in this category. Once the annotations were completed and verified, the dataset was exported in a YOLOv8-compatible format.

To facilitate analysis, the collected images were then structured and classified into four distinct categories: helmet, head, motorcycle, and person. The label distribution for each of these categories can be seen in Table 1. The number of images in each classification is part of the total number of images collected.

Table 1. Label distribution

ID	Class	Amount
0	Helmet	1.598
1	Head	1.866
2	Motorcycle	1.916
3	Person	1.231

The collected dataset was partitioned into training, validation, and testing. A total of 2.202 images, representing 70% of the entire dataset, were allocated for the training phase. Concurrently, 630 images, equivalent to 20% of the total samples, were designated for the validation process to optimize model parameters. The remaining 314 images, constituting 10% of the comprehensive dataset, were prepared for the testing stage to evaluate the model's performance objectively. Examples of the dataset can be seen in Figure 2.

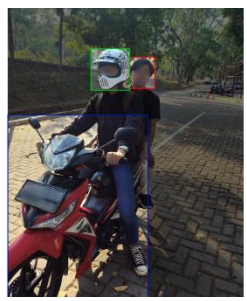


Figure 2. Example of a dataset

All image samples in this dataset were captured with careful consideration of varied shooting angles and diverse capture distances. This approach was implemented to ensure the dataset encompasses a broad diversity that authentically represents the dynamic and multifaceted situations encountered in the parking area environment. By integrating such comprehensive variability, the research aimed to develop a robust and adaptable helmet detection model capable of performing effectively across different real-world scenarios.

3.3 Object detection using YOLOv8

YOLOv8 implements an advanced neural network

architecture for high-precision real-time object detection [25]. The model integrates an efficient feature extraction process with a single-stage detection system, enabling rapid and accurate identification of objects such as vehicles and pedestrians [26].

YOLOv8 comprises an architecture consisting of three primary components, as shown in Figure 3 [27] that operate synergistically to generate accurate and efficient object detection: the Backbone serves as a feature extractor for input images through ConvModule and CSP-A, complemented by SPPF to capture multi-scale information [27]; the Neck is tasked with reinforcing extracted features using VoVGSCSP modules, Concat operations to combine features from various layers, Upsample processes to enhance feature resolution, and GSConv to optimize processing efficiency [27]; and the Head, responsible for object detection predictions, employs additional ConvModule and a final Conv2d layer to generate precise bounding box, classification, and confidence score predictions, with the integration of these three components enabling YOLOv8 to detect objects in real-time with high performance [27].

Utilizing the YOLOv8n model with the Ultralytics framework, this research leverages object detection architecture for a custom dataset. Implementation was conducted with training parameters including 50 epochs, a batch size of 16, and an input image resolution of 640×640 pixels. These parameters represent standard configurations commonly used in YOLOv8n model training, with adjustments to the epoch count to enable comprehensive learning from the custom dataset. While no advanced data augmentation techniques (e.g., flipping, rotation, or brightness adjustments) were applied, basic preprocessing steps such as Auto-Orient and Resize (Stretch to 640×640) were performed within Roboflow to standardize image orientation and conform to the input dimensions required by YOLOv8n.

3.4 Inference model in Hailo-8L

The Hailo-8L accelerator represents a significant advancement in edge computing technology for artificial intelligence applications. This low-power neural processing unit (NPU) is specifically designed to execute deep learning algorithms on edge devices and is compatible with various frameworks, including TensorFlow, PyTorch, Keras, and ONNX [23]. Hailo-8L offers computational performance up to 13 TOPS with a typical power consumption of only 1.5 watts, achieving an impressive efficiency of 8.7 TOPS/W [23].

Deploying the YOLOv8n model on Hailo-8L hardware requires a systematic conversion process, as illustrated in Figure 4 [28]. Initially, the trained PyTorch model undergoes ONNX conversion to establish a standardized intermediate representation. This ONNX model is subsequently processed through the Hailo Model Zoo and Dataflow Compiler, which optimizes the neural network architecture for the Hailo-8L's specialized hardware architecture. The optimization includes quantizing model weights and activation functions to 8-bit precision, restructuring memory layout, and operation fusion to maximize computational efficiency [29]. This conversion process preserves the model's detection accuracy while significantly reducing computational requirements, enabling real-time inference capabilities on the edge device.

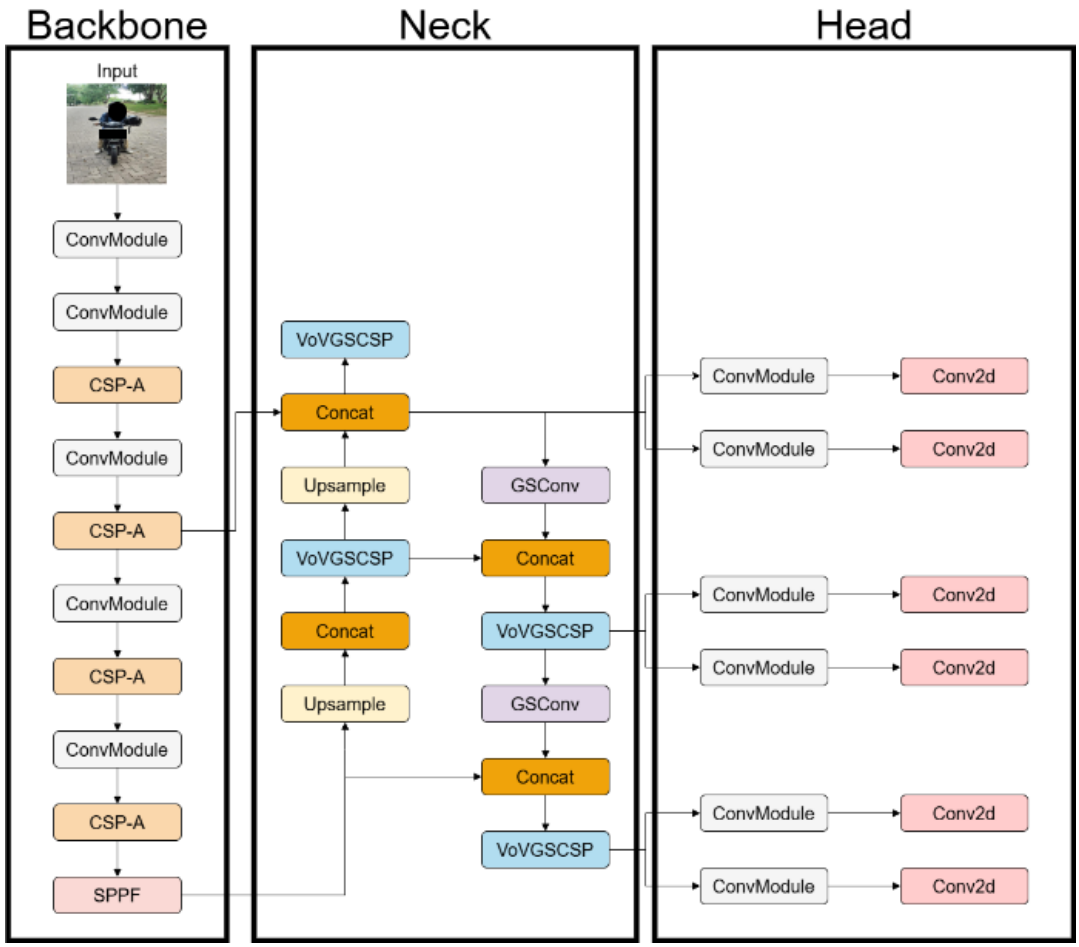


Figure 3. YOLOv8 architecture [27]

3.5 System integration

The block diagram in Figure 5 provides an overview of the system architecture for the automated barrier gate. It illustrates the interaction between various hardware components, including the Raspberry Pi 5 as the central processing unit, and the Arduino Uno for controlling multiple peripheral devices. The communication media used in Figure 5 is explained further in Table 2. The system incorporates a loop vehicle detector for vehicle presence detection, a USB camera for object recognition, and a linear actuator controlled via an L298N motor driver to operate the barrier gate. Additionally, power distribution is managed through a dual-source system: 220V AC and 12V DC. This structured integration enables efficient and reliable automation of the parking portal.

This system utilizes two power sources, 220V AC and 12V DC, to ensure that all components operate according to their respective requirements. The 220V AC power source supplies electricity to the Raspberry Pi 5, monitor, cooling fan, and Hailo-8L, while the 12V DC power source is used for the linear actuator through the L298N motor driver.

As the central processing unit, the Raspberry Pi 5 processes data obtained from the USB camera, which detects rider compliance. The Raspberry Pi 5 is equipped with a cooling fan to prevent overheating, while the Hailo-8L module accelerates machine learning-based computations.

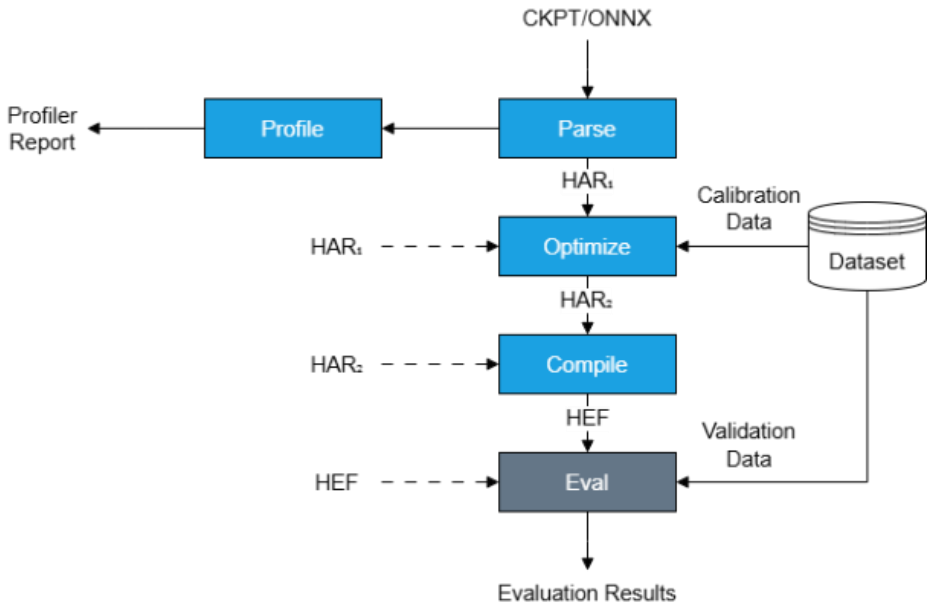


Figure 4. Model optimization and evaluation workflow [28]

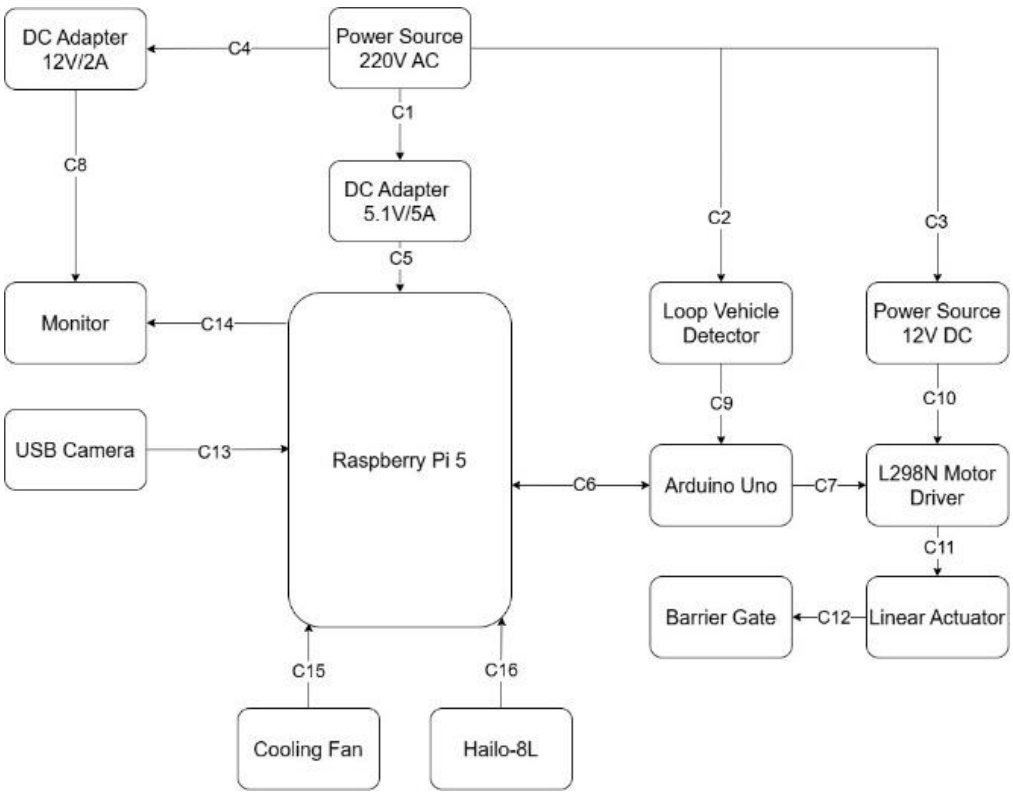


Figure 5. System block diagram

Table 2. System block diagram details

Code	Sender Components	Receiver Components	Communication Media
C1	Power source 220V AC	DC Adapter 5.1V/5A	Power cable
C2	Power source 220V AC	Loop vehicle detector	Power cable
C3	Power source 220V AC	Power source 12V DC	Power cable
C4	Power source 220V AC	DC Adapter 12V/2A	Power cable
C5	DC Adapter 5.1V/5A	Raspberry Pi 5	USB Type-C cable
C6	Raspberry Pi 5 Arduino Uno	Arduino Uno Raspberry Pi 5	Serial cable
C7	Arduino Uno	L298N motor driver	Jumper cable
C8	DC Adapter 12V/2A	Monitor	DC jack male-female
C9	Loop vehicle detector	Arduino Uno	Jumper cable
C10	Power source 12V DC	L298N motor driver	Power cable
C11	L298N motor driver	Linear actuator	Power motor cable
C12	Linear actuator	Barrier gate	Bolts & nuts
C13	USB camera	Raspberry Pi 5	USB Type-A cable
C14	Raspberry Pi 5	Monitor	HDMI cable
C15	Cooling fan	Raspberry Pi 5	Jumper cable
C16	Hailo-8L	Raspberry Pi 5	PCIe protocol

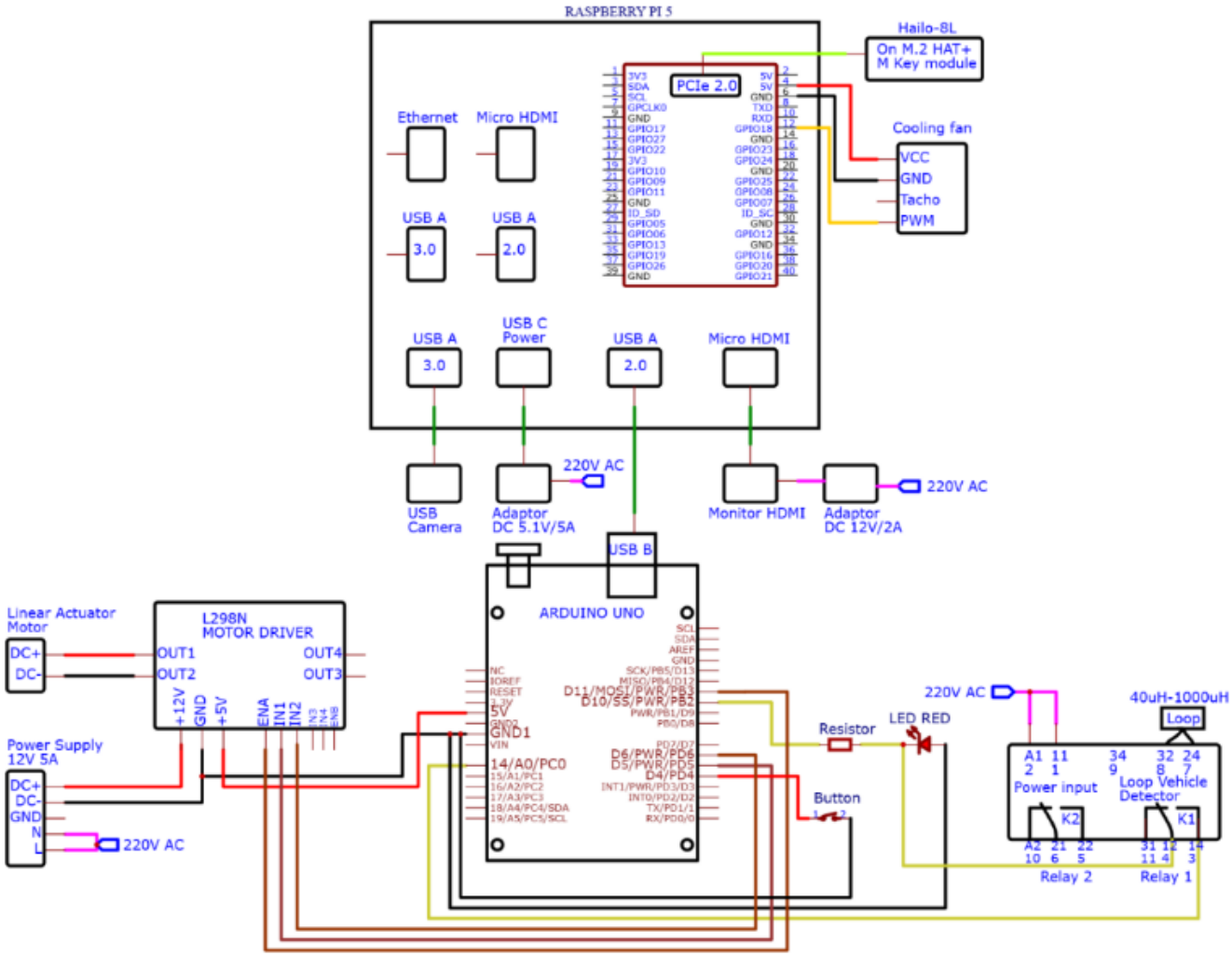


Figure 6. Schematic diagram

Additionally, the system includes a loop vehicle detector, which is connected to the Arduino Uno and functions to detect the presence of a vehicle beneath the barrier gate. The Arduino Uno receives signals from the Raspberry Pi 5 and the loop vehicle detector and transmits them to the L298N motor driver, which controls the linear actuator to automatically open and close the barrier gate.

The schematic diagram in Figure 6 provides a more detailed representation of the hardware architecture previously outlined in the block diagram. It visually clarifies the interconnections between components within the proposed systems.

The flowchart diagram in Figure 7 illustrates the operational workflow of the proposed helmet compliance enforcement system. The process begins when the system is powered on, initializing essential components through an autostart

configuration. This ensures that the system is immediately functional without requiring manual activation. The USB camera activates once the system is ready, serving as the primary input device for detecting approaching vehicles and riders.

At this stage, the deep learning model, which utilizes YOLO-based object detection, processes the real-time video feed to identify key elements: motorcycles, riders, and helmets. If a bike is detected but the rider is not wearing a helmet, access is denied, and the barrier gate remains closed to enforce compliance. On the other hand, if the system confirms that the rider meets the safety requirements, the barrier gate automatically opens, allowing entry.

Following this, a loop vehicle detector sensor monitors whether an object remains under the barrier gate. This is crucial for preventing accidental closures while a vehicle is still passing through. If the sensor detects an object, the gate remains open until the area is clear. Once the motorcycle has completely passed through, the gate closes automatically, and the system resets to detect the next approaching vehicle.

The design from Figure 8 illustrated physical representation when the barrier gate control container is placed on its frame. This design describes the physical rear shape of the barrier gate control container. The container is depicted in a horizontal position and has 2 holes for button access and air circulation. Then, the physical rear shape of the barrier gate control container is depicted in a vertical position. This container is a place to accommodate components. This container has mounting holes for linear actuator access and bolt retaining holes for barrier gate supports. This design describes the physical frame shape of the barrier gate control container. The frame functions as a physical support for the barrier gate control container when it is operated. This frame has a length of 1.35 meters. This height is used based on the measurement of the height of the motorcyclist when riding a motorbike; the result is an average height of 1.5 meters.

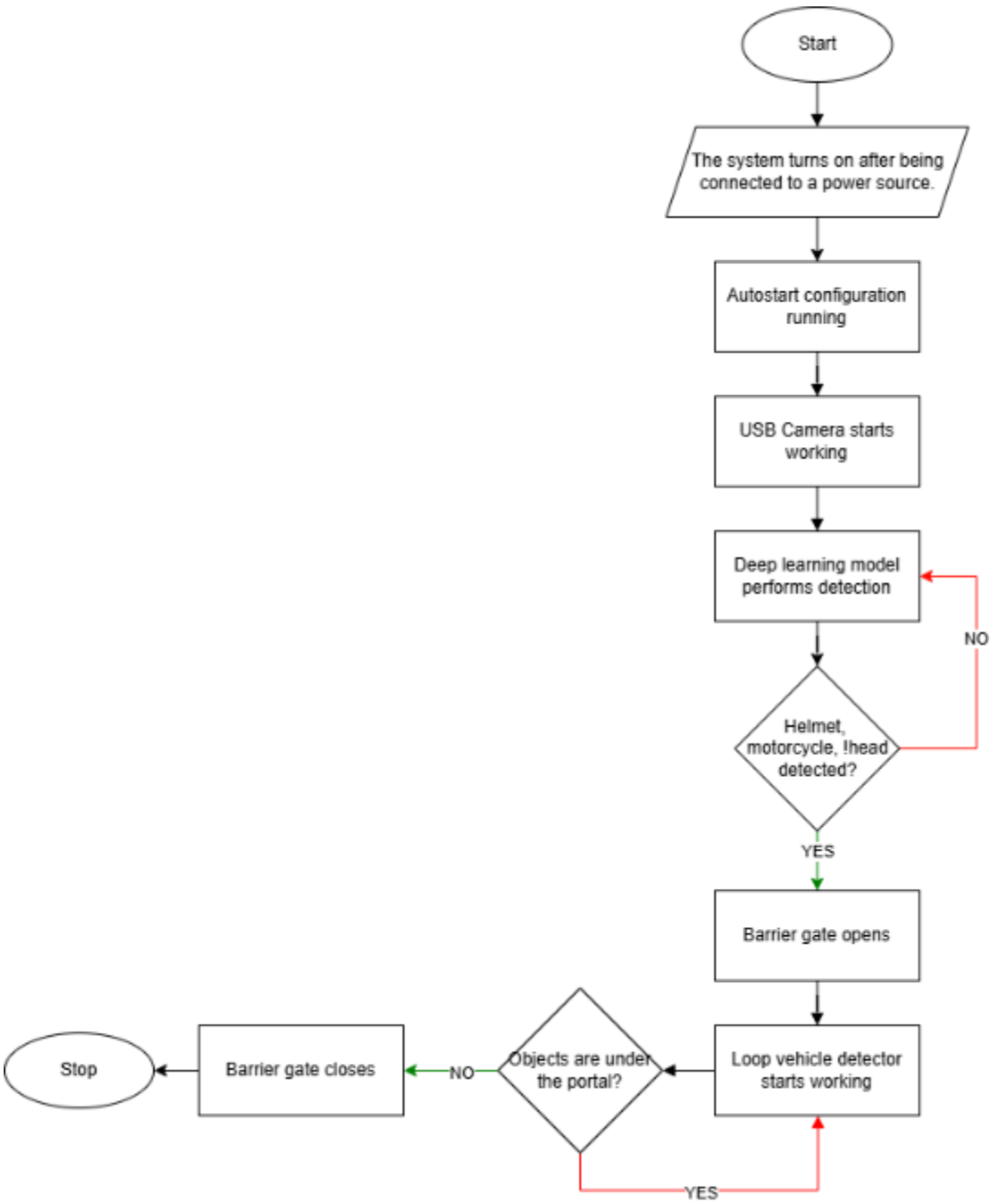
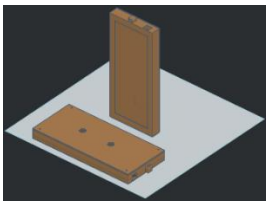
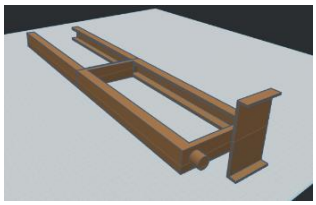


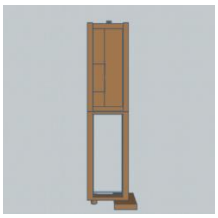
Figure 7. Flowchart diagram



Barrier gate container



Frame container



Barrier gate control

Figure 8. Hardware design

In the context of uniform motion, the relationship between displacement, speed, and time. Where s is the displacement or stroke length, v is the constant speed of the linear actuator, and t is the time taken. This fundamental concept of motion with constant speed is described in [30].

$$t = \frac{s}{v} \quad (1)$$

Analyzing the linear actuator's movement characteristics is essential to determining its maximum lifting time. The relationship between these parameters can be expressed mathematically, as shown in Eq. (1) [30].

$$t = \frac{Stroke\ length\ linear\ actuator}{Speed\ linear\ actuator}$$

Then,

$$t = \frac{0.2}{0.03} \approx 6.67\ seconds$$

Therefore, when supplied with an operational voltage of 12V DC, the linear actuator reaches its maximum extension in approximately 6.67 seconds. Meanwhile, the power supply was also increased to 18V to compare the speed. The results show that the time for the linear actuator to reach the maximum position is reduced to 5 seconds. However, because the size of the 18V power supply is too large to be placed in the barrier gate control container, the 12V DC is still used.

The linear actuator is connected at a point 0.12 meters from the left end of the portal. Therefore, the height change at that point can be calculated. When the linear actuator reaches its full stroke length of 0.2 meters, the crossbar connection point will be lifted by 0.2 meters. The relationship between the vertical and horizontal components in determining the inclination angle is a fundamental application of trigonometric functions, specifically the tangent function, as introduced in [31].

$$\theta = tan^{-1}\left(\frac{y}{x}\right) \quad (2)$$

Accordingly, the final angle of the crossbar can be calculated using the trigonometric equation as shown in Eq. (2) [31].

$$\theta = tan^{-1}\left(\frac{Linear\ actuator\ height\ increase}{Horizontal\ distance}\right)$$

The horizontal distance between the retaining bolt (0.2 meters from the end) and the actuator connection (0.12 meters from the end) is:

$$\theta = tan^{-1}\left(\frac{0.2}{0.1}\right)$$

Then,

$$\theta = tan^{-1}(2) \approx 63.43^{\circ}$$

So, when the linear actuator reaches its highest point, the bar will form an angle of about 63.43° from its initial position.

4. RESULTS AND ANALYSIS

The components contained in the barrier gate control container, such as Raspberry Pi 5 along with USB camera and monitor connections, Arduino Uno, linear actuator, L298N motor driver, loop vehicle detector, and power supply, work together to enable the automatic operation of the barrier gate. Figure 9 shows the detailed arrangement and wiring of these components.

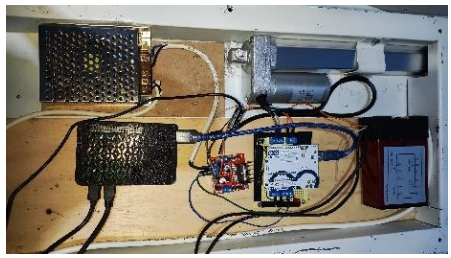


Figure 9. Barrier gate control components

Figure 10 shows the result of implementing the barrier gate control design for the automated barrier gate system. We used a 0.37-meter-long bar as a barrier gate and a 0.50-meter-long bar as a place to put the USB camera. In addition, we installed a monitor in front of the barrier gate container, which functions to display a preview of the helmet detection to the motorcyclist.

Figure 10. Barrier gate control

System response time testing is conducted to ensure the proposed system's reliability in detecting driving

completeness. This process consists of multiple sequential stages, starting with driver detection using a USB camera, data processing to assess driving completeness, and detection preview, culminating in the activation of the barrier gate to allow drivers a smooth passage. The test results are presented in Table 3.

Table 3. System response time

Iteration	System Response Time(s)
1	6,58
2	7,55
3	7,50
4	7,37
5	7
6	7,22
7	6
8	6,10
9	7,80
10	7

Based on the system response time test results, which have been carried out 10 iterations, the response time obtained ranges from 6 to 7.80 seconds with an average of 7.11 seconds.

Furthermore, the evaluation process was carried out using a test folder dataset containing 314 images specifically used for assessment. In object detection model testing, evaluation determines how well the model can classify objects correctly. This evaluation uses accuracy, precision, recall, and F1-score metrics. Additionally, a confusion matrix is used to analyze the model's performance in detecting each class in more detail, such as helmets, heads, motorcycles, and persons.

Accuracy measures how well a model makes correct predictions overall by comparing the number of correct predictions to the total predictions made. This research uses a confusion matrix to evaluate how effectively the system identifies aggressive and non-aggressive actions, as shown in Eq. (3) [32].

$$Accurary = \frac{(TN + TP)}{(TN + TP + FP + FN)} \tag{3}$$

Precision represents the proportion of correctly identified positive instances among all predicted positives, reflecting the model's accuracy in classifying positive samples, as defined in Eq. (4) [33].

$$Precision = \frac{TP}{(TP + FP)} \tag{4}$$

Recall, also known as the check-all rate, measures the model's ability to identify all actual positive instances correctly. It assesses how well the model captures true positives, as shown in Eq. (5) [33].

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

The F1-Score is the harmonic mean of precision and recall, offering a balanced evaluation of both metrics. A higher F1 score indicates better overall performance in maintaining a trade-off between precision and recall, as defined in Eq. (6) [33].

$$F1 - Score = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)} \tag{6}$$

The mean Average Precision (mAP) represents the average of the Average Precision (AP) across all object classes, providing a comprehensive measure of detection accuracy. A higher mAP indicates better performance in detecting objects across multiple courses, as defined in Eq. (7) [34].

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{7}$$

Based on Figure 11, the confusion matrix shows that while the model performs well in detecting head, helmet, motorcycle, and person classes, a significant number of instances were misclassified as “none”, meaning no object was detected. This issue occurred most frequently in the motorcycle class (48 cases), followed by head (30), helmet (21), and person (22). These misclassifications are caused by visual ambiguity, poor lighting conditions, and complex backgrounds that resemble the target objects [17].

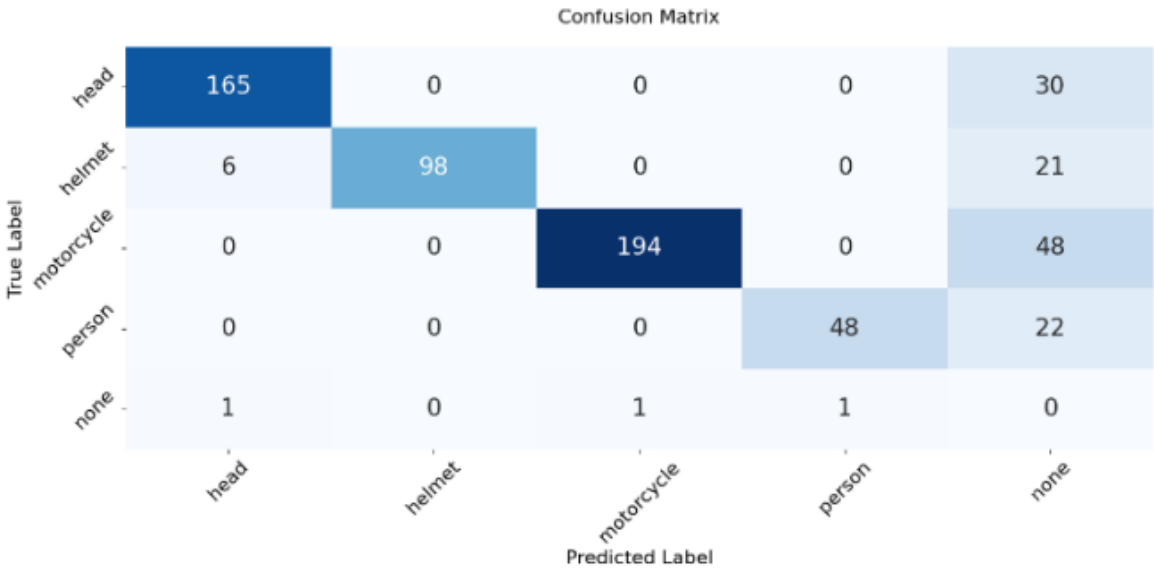


Figure 11. Confusion matrix visualization

To address this issue, several improvements are proposed. Applying data augmentation techniques such as brightness variation, rotation, and flipping can help the model adapt to diverse real-world conditions. Enhancing the dataset, particularly by adding more samples for the helmet and head classes in varied environments, is expected to improve model robustness. Adjusting the inference threshold from 0.7 to a slightly lower value, such as 0.6 or 0.5, can reduce false negatives by allowing more flexible detections. In addition, fine-tuning the YOLOv8n model or switching to a more capable variant like YOLOv8s, as well as exploring hybrid detection approaches [22], can further enhance feature extraction and overall detection accuracy. These optimizations aim to reduce "none" classifications and improve the system's reliability in real-world applications.

Based on the confusion matrix in Figure 11, the model's accuracy can be calculated by dividing the number of correct predictions by the total number of predictions, as shown in Eq. (3). The correct predictions are located on the main diagonal of the confusion matrix, which are 165 (head), 98 (helmet), 194 (motorcycle), and 48 (person), resulting in a total of 505 correct predictions. Meanwhile, the total number of predictions made by the model is 635.

$$Accuracy = \frac{165 + 98 + 194 + 48}{635}$$

Then,

$$Accuracy = \frac{505}{635} \approx 0.7952$$

The accuracy value of 0.7952 was obtained by calculating the ratio of correct predictions to the total number of predictions made. This value shows the model's ability to classify the test data correctly.

Table 4. Model performance metrics for each category

Performance Metrics	Helmet	Head	Motorcycle	Person
Precision	1.00	0.96	0.99	0.98
Recall	0.78	0.84	0.80	0.69
F1-Score	0.88	0.90	0.89	0.81

Based on Table 4, the confusion matrix analysis shows the system's reliable detection performance. The system achieves high precision across all categories: 1.00 for helmet, 0.96 for head, 0.99 for motorcycle, and 0.98 for person. The recall values indicate good detection performance, with 0.78 for helmet, 0.84 for head, 0.80 for motorcycle, and 0.69 for person. Furthermore, the F1-scores support the model's reliability, reaching 0.88 for helmet, 0.90 for head, 0.89 for motorcycle, and 0.81 for person.

Based on Table 4, which presents the Precision, Recall, and F1-Score values for each class, the mean Average Precision (mAP) is calculated by averaging the Precision values of each class: Helmet (1.00), Head (0.96), Motorcycle (0.99), and Person (0.98). These Precision values are used directly because the evaluation used a single confidence threshold, set at 0.7. With only one threshold, generating a complete Precision-Recall curve to compute the area under the curve is impossible. Therefore, the Precision at this specific threshold serves as a representation of the model's detection performance. This approach is commonly used when evaluation is performed at a single threshold point. The mAP is then calculated using Eq. (7), representing the average of the Precision values across all classes.

$$mAP = \frac{1}{4}(1.00 + 0.96 + 0.99 + 0.98)$$

Then,

$$mAP = 0.9825$$

Based on the calculated average of the Precision values for each class, the mean Average Precision (mAP) obtained is 0.9825. This value indicates that the model demonstrates a strong detection performance in identifying objects within the test data.

The results can be seen in Figure 12. The following is the implementation of the model presented to users in the detection application. In this display, detected objects such as helmets, heads, motorcycles, and people are enclosed within bounding boxes to indicate the model's prediction results. These bounding boxes allow users to observe how the model identifies and classifies each object in the image in real time.

Rider without helmet

Rider with helmet

Pedestrian

Figure 12. Sample detection results from the model

The graph presented in Figure 13 compares CPU usage between two configurations: with and without the Hailo-8L AI accelerator. The y-axis represents memory usage in percentage, while the x-axis denotes measurement time in seconds. The results indicate that without the Hailo-8L, the CPU utilization remains consistently high, averaging around 90%. In contrast, with the Hailo-8L, CPU usage is significantly reduced to approximately 20%, demonstrating the efficiency of the AI accelerator in offloading computational workloads. This reduction in CPU usage suggests improved processing efficiency when utilizing the Hailo-8L.

The graph presented in Figure 14 compares memory usage between two configurations: with and without the Hailo-8L AI accelerator. The y-axis represents memory usage in

percentage, while the x-axis denotes measurement time in seconds. The results show that without the Hailo-8L, memory usage remains slightly higher, averaging around 21%. In contrast, with the Hailo-8L, memory usage is reduced to approximately 18%. This indicates that utilizing the Hailo-8L reduces CPU load and optimizes memory consumption, contributing to overall system efficiency.

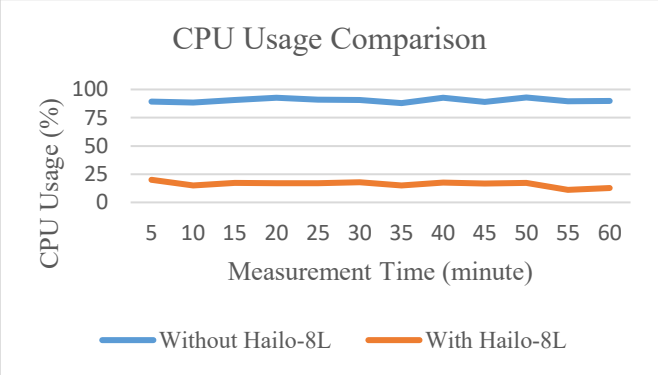


Figure 13. CPU usage comparison

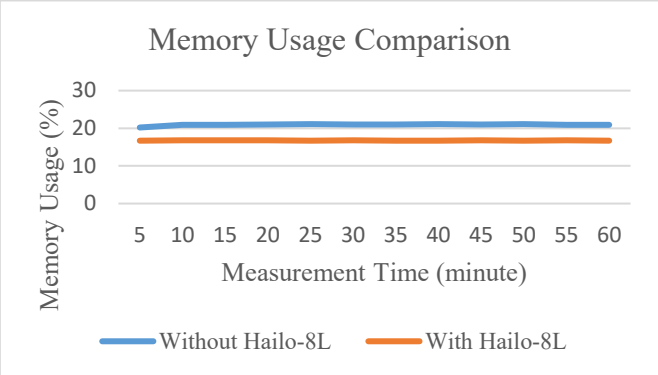


Figure 14. Memory usage comparison

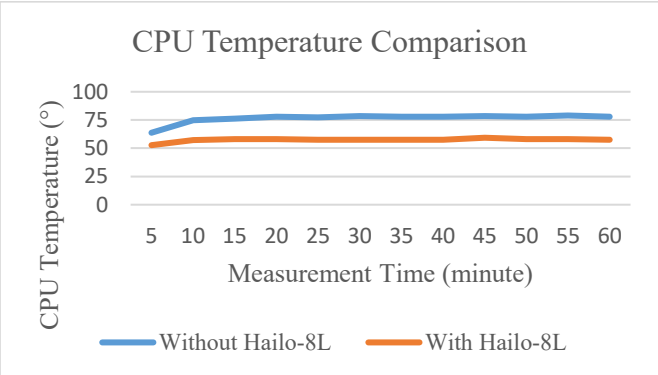


Figure 15. CPU temperature comparison

The graph presented in Figure 15 compares CPU temperature between two configurations: with and without the Hailo-8L AI accelerator. The y-axis represents CPU temperature in degrees Celsius, while the x-axis indicates measurement time in seconds. The results show that without the Hailo-8L, CPU temperature remains higher, stabilizing around 80°C. In contrast, when using the Hailo-8L, CPU temperature is noticeably lower, averaging around 65°C. This indicates that the Hailo-8L offloads computational tasks, reducing CPU workload and consequently lowering its operating temperature, which contributes to improved thermal efficiency and system stability.

Assuming the Raspberry Pi 5's CPU consumes approximately 5W at full load, the actual power consumption at a given moment can be estimated by multiplying this value by the average CPU utilization percentage, which is then calculated using Eq. (8).

$$P_{CPU} = 5W \times \frac{Average}{100} \tag{8}$$

CPU power without Hailo-8L:

$$P_{CPU} = 5W \times \frac{90.35}{100} = 4.5W$$

CPU power with Hailo-8L:

$$P_{CPU} = 5W \times \frac{16.19}{100} = 0.80W$$

Meanwhile, Hailo-8L typically consumes only 1.5 watts [23]. Thus, the total power when using Hailo is:

$$P_{Total} = P_{CPU} + P_{Hailo} = 0.80W + 1.5W = 2.3W$$

Hailo-8L on the Raspberry Pi 5 significantly reduces CPU and memory workload, power consumption, and system temperature. Hailo-8L handles machine learning tasks efficiently with a more optimized architecture than CPUs, which are not specifically designed for intensive machine learning. As a result, the system becomes more efficient, allowing use in deep learning applications without the risk of overheating that can significantly reduce throughput during long-term continuous inference, with further reductions if the ambient temperature is high [35]. Compared to the total power without Hailo-8L (around 4.5W), using Hailo-8L provides a power saving of 48.89%.

5. CONCLUSION

This research successfully demonstrates the implementation of an automated barrier gate by integrating several technologies, including Arduino for the barrier gate control system [8, 9], the Raspberry Pi as the processing unit for machine learning tasks [7, 10], YOLO as the object detection model [7], and the Hailo-8 AI accelerator to enhance inference efficiency [11].

The confusion matrix analysis results indicate that the system demonstrates good detection performance, as reflected by a mean Average Precision (mAP) of approximately 0.9825 for all classes (helmet, head, motorcycle, and person). However, some improvements are still needed. The current system has limitations, such as its reliance on good lighting conditions for accurate image processing. Using a higher-quality camera module or integrating IR-assisted imaging could improve robustness under poor lighting. Additionally, during peak traffic hours, detection time slightly increases due to hardware processing limits. Increasing the motor driver voltage to 18V was found to reduce the average response time to 5 seconds. Another critical limitation is weather sensitivity—the system is currently not rainproof, making it unsuitable for operation during heavy rain. This can be

addressed by using waterproof enclosures and conducting tests under various weather conditions. Future developments should also explore optimizing detection algorithms, enhancing hardware integration, and expanding the system for broader smart environment applications. Improving model training with more varied datasets.

The proposed system presents a novel integration of deep learning processing using the Raspberry Pi 5 and automated hardware control via the Arduino Uno to detect motorcyclist compliance. It builds upon and enhances previously independent systems by combining them into a unified architecture. The system employs the YOLOv8n object detection algorithm, which offers fast and accurate real-time recognition of vehicles and riders with a lightweight design optimized for edge computing. Integrated with the Hailo-8L AI accelerator, YOLOv8n enables more efficient inference by reducing the computational workload on the Raspberry Pi 5. Experimental results demonstrate that the Hailo-8L significantly reduces CPU usage, memory consumption, and operating temperature. It also lowers power consumption, achieving a power saving of up to 48.89%.

AUTHOR CONTRIBUTION

Agustinus Adven Christo is the Hardware Developer of this research and also contributed as the author of this manuscript. M. Bintang Prayoga Utama and Yosia Aser Camme are the Artificial Intelligence Developers of this research and contributed as the authors of this manuscript. Dania Eridani, the research supervisor, contributed as the author of this manuscript and the corresponding author. Patricia Evericho Mountaines, the research supervisor, contributed as the author of this manuscript.

REFERENCES

[1] Badan Pusat Statistik. Development of the Number of Motor Vehicles by Type (Unit), 2021-2022. Badan Pusat Statistik. https://www.bps.go.id/id/statistics-table/2/NTcjMg==/perkembangan-jumlah-kendaraan-bermotor-menurut-jenis--unit-.html, accessed on Apr. 09, 2025.

[2] Badan Pusat Statistik. Number of Accidents, Fatalities, Serious Injuries, Minor Injuries, and Material Losses, 2022. Badan Pusat Statistik. https://www.bps.go.id/id/statistics-table/2/NTEzIzI=/jumlah-kecelakaan--korban-mati--luka-berat--luka-ringan--dan-kerugian-materi.html, accesed on Apr. 09, 2025.

[3] Mercado Reyna, J., Luna-Garcia, H., Espino-Salinas, C.H., Celaya-Padilla, J.M., Gamboa-Rosales, H., Galván-Tejada, J.I., Solís Robles, R., Rondon, D., Villalba-Condori, K.O. (2023). Detection of helmet use in motorcycle drivers using convolutional neural network. Applied Sciences, 13(10): 5882. https://doi.org/10.3390/app13105882

[4] Utami, P.V. (2024). The application of technology and information development in electronic traffic law enforcement (ETLE) to shape public awareness. Edusight International Journal of Multidisciplinary Studies, 1(1). https://doi.org/10.69726/eijoms.v1i1.9

[5] Fahim, A., Hasan, M., Chowdhury, M.A. (2021). Smart parking systems: Comprehensive review based on various aspects. Heliyon, 7(5): e07050. https://doi.org/10.1016/j.heliyon.2021.e07050

[6] Malik, I., Verma, S., Sachan, S. (2024). YOLOv7-based helmet detection system for traffic safety. In 2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, pp. 493-497. https://doi.org/10.1109/CCICT62777.2024.00082

[7] Charef, A., Jarir, Z., Quafafou, M. (2024). Enhancing road safety: Automated traffic violation detection and counting system using YOLO algorithm. In 2024 Mediterranean Smart Cities Conference (MSCC), Martil - Tetuan, Morocco, pp. 1-6. https://doi.org/10.1109/MSCC62288.2024.10697076

[8] Widodo, S., Amin, M.M., Sutrisman, A., Cofriyanti, E., Puji, R.M. (2020). Implementation of parking portal door security system using RFID and password based on microcontrollers in Sriwijaya state polytechnic. Journal of Physics: Conference Series, 1500(1): 012114. https://doi.org/10.1088/1742-6596/1500/1/012114

[9] Dixit, M., Priya, A., Haldiya, G., Priya, A., Kumar, B. (2023). Smart car parking system using Arduino. In 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India, pp. 1-6. https://doi.org/10.1109/SCEECS57921.2023.10063121

[10] Mala, S., Vidyashree, H.R., Chanda, K. (2024). Yolo model-based license plate extraction and toll generation for smart parking systems. In 2024 2nd International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, pp. 1-7. https://doi.org/10.1109/ICNEWS60873.2024.10730931

[11] Achmadiah, M.N., Setyawan, N., Bryantono, A.A., Sun, C.C., Kuo, W.K. (2024). Fast person detection using YOLOX with AI accelerator for train station safety. In 2024 International Electronics Symposium (IES), Denpasar, Indonesia, pp. 504-509. https://doi.org/10.1109/IES63037.2024.10665874

[12] Kondaveeti, H.K., Kumaravelu, N.K., Vanambathina, S.D., Mathe, S.E., Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. Computer Science Review, 40: 100364. https://doi.org/10.1016/j.cosrev.2021.100364

[13] Mathe, S.E., Kondaveeti, H.K., Vappangi, S., Vanambathina, S.D., Kumaravelu, N.K. (2024). A comprehensive review on applications of Raspberry Pi. Computer Science Review, 52: 100636. https://doi.org/10.1016/j.cosrev.2024.100636

[14] Hermawan, S., Rochardjo, H.S. (2022). Preliminary design of electric linear actuator for hospital bed domestic product. Journal of Mechanical Design and Testing, 4(1): 25-31. https://doi.org/10.22146/jmdt.63146

[15] Nava-Pintor, J.A., Carlos-Mancilla, M.A., Guerrero-Osuna, H.A., Luque-Vega, L.F., Carrasco-Navarro, R., Castro-Tapia, S., Mata-Romero, M.E., González-Jiménez, L.E., Solís-Sánchez, L.O. (2023). Design, implementation, and control of a linear electric actuator for educational mechatronics. Machines, 11(9): 894. https://doi.org/10.3390/machines11090894

[16] Kumar, K., Singh, V., Raja, L., Bhagirath, S.N. (2023). A review of parking slot types and their detection

techniques for smart cities. Smart Cities, 6(5): 2639-2660. https://doi.org/10.3390/smartcities6050119
[17] Kim, B.I., Ko, B.C., Jang, I.S., Kim, K.J. (2024). Helmet detection of motobike riders in real-world scenarios. In 2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Danang, Vietnam, pp. 1-4. https://doi.org/10.1109/ICCE-Asia63397.2024.10773688
[18] Taye, M.M. (2023). Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. Computers, 12(5): 91. https://doi.org/10.3390/computers12050091
[19] Gallo, G., Di Rienzo, F., Garzelli, F., Ducange, P., Vallati, C. (2022). A smart system for personal protective equipment detection in industrial environments based on deep learning at the edge. IEEE Access, 10: 110862-110878. https://doi.org/10.1109/ACCESS.2022.3215148
[20] Abirami, M.S., Jain, H.K., Kanwar, A.S. (2024). Detection of two-wheelers traffic violations and automated ticketing using YOLOv8. In 2024 International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET), Indore, India, pp. 1-6. https://doi.org/10.1109/ACROSET62108.2024.10743298
[21] Chai, Z. (2024). Real-time automatic detection of motorcycle helmet based on improved YOLOv8 algorithm. In 2024 6th International Conference on Communications, Information System and Computer Engineering (CISCE), Guangzhou, China, pp. 1239-1243. https://doi.org/10.1109/CISCE62493.2024.10652610
[22] Vaikunth, M., Dejey, D., Vishaal, C., Balamurali, S. (2024). Optimizing helmet detection with hybrid YOLO pipelines: A detailed analysis. In CS & IT Conference Proceedings, pp. 83-93. https://doi.org/10.5121/csit.2024.142406
[23] Bueno, G., Sanchez-Vargas, L., Diaz-Maroto, A., Ruiz-Santaquiteria, J., Blanco, M., Salido, J., Cristobal, G. (2025). Real-time edge computing vs. GPU-accelerated pipelines for low-cost microscopy applications. Electronics, 14(5): 930. https://doi.org/10.3390/electronics14050930
[24] Luculescu, M.C., Cristea, L., Boer, A.L. (2025). Artificial vision system for autonomous mobile platform used in intelligent and flexible indoor environment inspection. Technologies, 13(4): 161. https://doi.org/10.3390/technologies13040161
[25] Zhou, Q., Wang, Z., Zhong, Y., Zhong, F., Wang, L. (2024). Efficient optimized YOLOv8 model with extended vision. Sensors, 24(20): 6506. https://doi.org/10.3390/s24206506
[26] Liu, W., Qiao, X., Zhao, C., Deng, T., Yan, F. (2025). VP-YOLO: A human visual perception-inspired robust vehicle-pedestrian detection model for complex traffic scenarios. Expert Systems with Applications, 274: 126837. https://doi.org/10.1016/j.eswa.2025.126837
[27] Chen, Z., Liu, B., Gao, X., Sun, S., Yao, B. (2024). Safety helmet detection using improved YOLOv8. In 2024 5th International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), Wenzhou, China, pp. 677-682. https://doi.org/10.1109/ICBASE63199.2024.10762189
[28] Hailo Technologies Ltd., hailo_model_zoo, GitHub. https://github.com/hailo-ai/hailo_model_zoo, accessed on Apr. 11, 2025.
[29] Pohl, D., Vogel-Heuser, I.B., Krüger, M., Echtler, M. (2024). Quantization effects of deep neural networks on a FPGA platform. In 2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS), St. Louis, MO, USA, pp. 1-8. https://doi.org/10.1109/ICPS59941.2024.10640013
[30] Paul Peter Urone and Roger Hinrichs, Physics. Houston: OpenStax, 2020. https://openstax.org/books/physics/pages/1-introduction, accessed on Apr. 12, 2025.
[31] Paul Peter Urone and Roger Hinrichs, College Physics 2e. Houston: OpenStax, 2022. https://openstax.org/books/college-physics-2e/pages/1-introduction-to-science-and-the-realm-of-physics-physical-quantities-and-units, accessed on Apr. 12, 2025.
[32] Wu, M.T. (2022). Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom. Scientific Reports, 12(1): 3095. https://doi.org/10.1038/s41598-022-07137-z
[33] Cheng, L. (2024). A highly robust helmet detection algorithm based on YOLO V8 and Transformer. IEEE Access. 12: 130693-130705. https://doi.org/10.1109/ACCESS.2024.3459591
[34] Padilla, R., Netto, S.L., Da Silva, E.A. (2020). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP), Niteroi, Brazil, pp. 237-242. https://doi.org/10.1109/IWSSIP48289.2020.9145130
[35] Benoit-Cattin, T., Velasco-Montero, D., Fernández-Berni, J. (2020). Impact of thermal throttling on long-term visual inference in a CPU-based edge device. Electronics, 9(12): 2106. https://doi.org/10.3390/electronics9122106

NOMENCLATURE

t	time, s
s	distance, m
v	speed, m/s
y	vertical component, m
x	horizontal component, m
P	power, W
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
AP	Average Precision
mAP	Mean Average Precision

Greek symbols

θ	angle, °

Subscripts

CPU	Central Processing Unit
Hailo	Hailo Accelerator Hardware