IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 99, No. 1, Month 2099, pp. 1~1x
ISSN: 2252-8938, DOI: 10.11591/ijai.v99.i1.pp1-1x

Optimized Real-Time Object Detection on Raspberry Pi 5 Using YOLOv5n/Openvino for VTOL

Ryan Satria Wijaya[1], Astuti Yeirene Panggabean[2]
[1]Robotic Engineering, Electrical Engineering, Politeknik Negeri Batam, Batam City, Indonesia [2]Robotic Engineering, Electrical Engineering, Politeknik Negeri Batam, Batam City, Indonesia

Article Info

Article history:

Received month dd, yyyy
Revised month dd, yyyy
Accepted month dd, yyyy

Keywords:

Raspberry Pi 5
YOLOv5n
OpenVINO IR
Real-time object detection
VTOL

ABSTRACT

This study evaluates a real-time object detection system on Raspberry Pi 5 for vision-based autonomous navigation of VTOL drone platforms. A lightweight YOLOv5n model was trained at 640×640 pixels and optimized by converting it to OpenVINO Intermediate Representation (IR) format with a reduced input size of 416×416 pixels to improve CPU inference speed on edge hardware. The dataset consists of three object classes (baskets, cubes, and tori) comprising 4,575 images with 10,260 annotated instances (approximately 3,420 annotations per class), collected in various environmental conditions using the Roboflow platform. Experiments were conducted on a VTOL drone platform under controlled and tethered conditions using a USB camera connected to the Raspberry Pi 5. Results show the system achieves mAP@0.5 of around 0.992 and a maximum F1-score of around 0.992 at a threshold of 0.43, while the confusion matrix indicates high classification accuracy across all classes. The system achieves an average throughput of approximately 18 FPS and inference time of approximately 55 ms per frame across 2,160 outdoor test frames, with moderate CPU and memory usage. This demonstrates that YOLOv5n with OpenVINO on the Raspberry Pi 5 enables high-performance real-time multi-class detection, providing a strong foundation for drone navigation systems development.

Corresponding Author:

Ryan Satria Wijaya
Robotics Engineering, Electrical Engineering Politeknik Negeri Batam
Batam City, Riau Island, Indonesia
Email: ryan@polibatam.ac.id

1. INTRODUCTION

High-performance real-time object identification based on deep learning has become an important component of many security systems, including robotics, unmanned aerial vehicle (UAV) applications, and surveillance [1]. The YOLO (You Only Look Once) model has been widely adopted due to its ability to balance accuracy and inference speed, making it ideal for deployment on advanced computing devices[2]. Recent studies have demonstrated that YOLO variants, particularly YOLOv5 through YOLOv8, can be optimized to create lightweight models without significant degradation in detection capabilities, making them highly relevant for vision-based navigation systems that require real-time visual decisions with minimal latency and low power consumption[3], [4]. The integration of object detection modules on autonomous platforms, such as VTOL (Vertical Take-Off and Landing) drones, enables critical usage scenarios including automatic target landing, terrain mapping, and vision-based navigation[5], [6]. However, the trade-off between computational capacity

and battery life necessitates highly efficient and lightweight detection algorithms for mobile autonomous systems[7]. The Raspberry Pi 5, as the latest generation of single-board computers, offers significant improvements with its quad-core 2.4 GHz Cortex-A76 CPU and memory capacity up to 8 GB LPDDR4X compared to previous models[8], making it suitable for computer vision applications at the edge[9]. Recent benchmarking studies have shown that the Raspberry Pi 5 can achieve substantial performance improvements for object detection tasks compared to earlier generations[10].

This configuration is particularly relevant for robotic systems and unmanned aerial vehicles (UAVs), as optimization through inference frameworks such as OpenVINO enables accelerated model execution on ARM CPUs without the need for specialized hardware accelerators[11]. OpenVINO optimization capabilities, including model conversion to Intermediate Representation (IR) format and runtime inference acceleration, have been demonstrated to significantly improve inference performance on edge devices[12].

Despite this progress, several critical gaps remain in the application of YOLOv5n for autonomous navigation on edge devices. First, existing benchmarks focus on older generations of Raspberry Pi such as Raspberry Pi 3 and Raspberry Pi 4 [13] or comparative YOLO studies [14] without fully exploring the enhanced capabilities of Cortex-A76 on Raspberry Pi 5 for YOLOv5n inference[15]. Second, although OpenVINO optimizations have been studied[16], empirical evidence on YOLOv5n conversion with multi-resolution strategies (640×640 vs. 416×416) remains limited[17].

By capitalizing on these gaps, this study specifically targets the under-explored combination of the Raspberry Pi 5 Cortex-A76 architecture and the dual-resolution YOLOv5n training-to-deployment strategy (640×640 pixels for training, 416×416 pixels for OpenVINO IR inference), evaluated through real-world tethered VTOL flights rather than just static benchmarking. Unlike previous studies that reported on YOLO-OpenVINO combinations primarily through offline accuracy metrics or on older generations of Raspberry Pi [13]–[17], this research combines model-level accuracy evaluation with real-time outdoor benchmarking (FPS, inference time, and CPU utilization across 2,160 frames) on an actual VTOL platform, which constitutes a specific novelty and practical contribution of this study.

This research focuses on the development and evaluation of a multi-class object detection system using the YOLOv5n model on a Raspberry Pi 5 with 8 GB of RAM[18]. The model was initially trained at a resolution of 640×640 pixels and subsequently converted to OpenVINO Intermediate Representation (IR) format with an optimized resolution of 416×416 pixels to improve computational efficiency[19]. Accuracy performance was evaluated using comprehensive metrics including precision, recall, F1-score, mAP@0.5, mAP@0.5:0.95, and confusion matrix analysis[20], while real-time performance was measured through FPS, inference time, CPU utilization, and memory consumption across test frames[21].

This framework is designed to meet the requirements of vision-based autonomous navigation applications, particularly on platforms with limited computational resources such as UAVs and mobile robots[22], [23]. The main contribution of this research is the demonstration of a comprehensive pipeline encompassing YOLOv5n training, OpenVINO optimization, and real-time implementation with quantitative performance analysis on the Raspberry Pi 5[24]. The experimental results demonstrate that the system achieves multi-class detection with a mAP@0.5 of approximately 0.992 and an average throughput of approximately 18 frames per second in real-time testing, which is suitable for vision-based autonomous navigation applications[25]. Additionally, this research presents a comprehensive analysis of FPS distribution, latency variation, and resource utilization[26], while identifying computational bottlenecks and proposing potential future optimizations such as INT8 quantization, model assembling, and adaptive resolution strategies as avenues for advancement[27].

2. METHOD

This research consists of two distinct phases conducted on different hardware platforms: offline training on a workstation with high processing power and online inference on an edge device (Raspberry Pi 5). The system is designed for onboard deployment on a VTOL drone platform, where experiments are conducted under controlled and tethered conditions. The hardware specifications for both phases are listed in Table 1.

Table 1. Hardware specifications for training and deployment

Component	Specifications for Training	Deployment Edge
CPU	AMD Ryzen 5 5600X (3.7 GHz x 12)	Raspberry Pi 5 (8GB LPDDR4X RAM, Quad-core Arm Cortex-A76 2.4GHz)
GPU	NVIDIA GeForce RTX	-

	3060, 12 GB	
RAM	16 GB DDR4×2 (32 GB Dual Channel)	LPDDR4X 8 GB
OS	Ubuntu 22.04 LTS (64-bit)	Raspberry Pi OS 64-bit
Framework	Torch 2.0.1 + Torch Vision 0.15.2	OpenVINO™ 2023.0.0
Storage	SSD 500 GB + HDD 2 TB	Micro SD 32 GB
Accelerator	OpenVINO™ Runtime (CPU inference)	OpenVINO™ Runtime on ARM Cortex-A76 (no dedicated accelerator)

Figure 1 illustrates the entire research process from start to finish, including data collection and analysis, model analysis and accuracy evaluation, conversion of the YOLOv5n model to OpenVINO Intermediate Representation (IR), and implementation and real-time performance on the Raspberry Pi 5 platform.

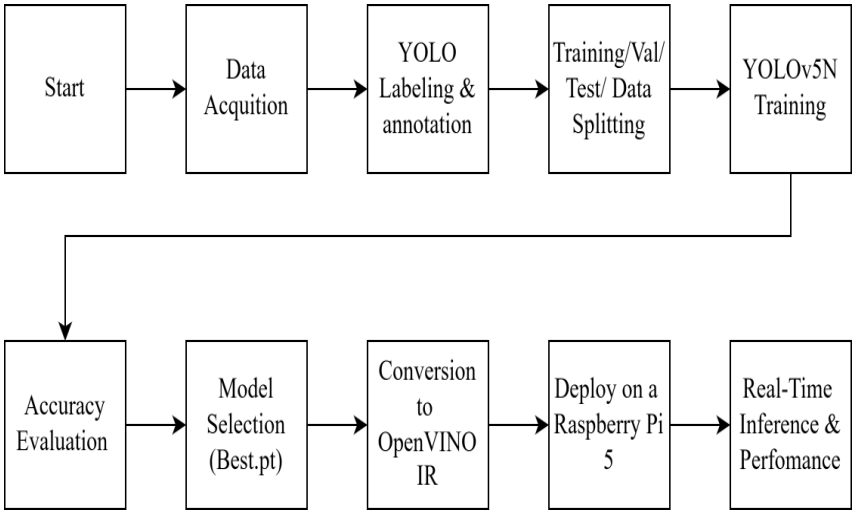

Figure 1. Overall research workflow.

The work began with dataset analysis and manual data entry using the YOLO format, followed by training-validation-testing and data augmentation. The YOLOv5n model was then analyzed and evaluated using standard object detection methods. The best-performing model was converted to OpenVINO IR format and deployed on a Raspberry Pi 5 to measure time using a USB camera, where accuracy and resource usage were measured within the frame. This design ensures that the model is not only accurate for test data, but also useful for time-sensitive operations on edge devices with limited computing resources.

2.1 Dataset and data preparation

In Figure 2 below, this dataset consists of three types of objects, namely baskets, cubes, and tori, which were collected indoors and outdoors at varying distance and lighting conditions. There are a total of 4,575 images with resolutions of 1280x720 and 1920x1080 pixels used. Each image may contain more than one object, resulting in a total of 10,260 annotated instance (approximately 3,420 annotations per class). Each image is manually annotated using the Bounding Box tool (Roboflow), which uses the normalized YOLO coordinate format (x_{center}, y_{center}, *width*, *height*) for each object instance. The annotated data set is then divided into 70% training data, 20% validation data, and 10% test data to support proper model generalization. To capture scene diversity, images were gathered from various indoor and outdoor settings, with camera-to-object distances ranging from approximately 0.5 to 5 meters and different viewing angles, while maintaining a nearly balanced class distribution of 673, 689, and 680 test instances for the basket, cube, and torus classes, respectively (Table 2), in order to mitigate the potential for class-imbalance bias during training and assessment.

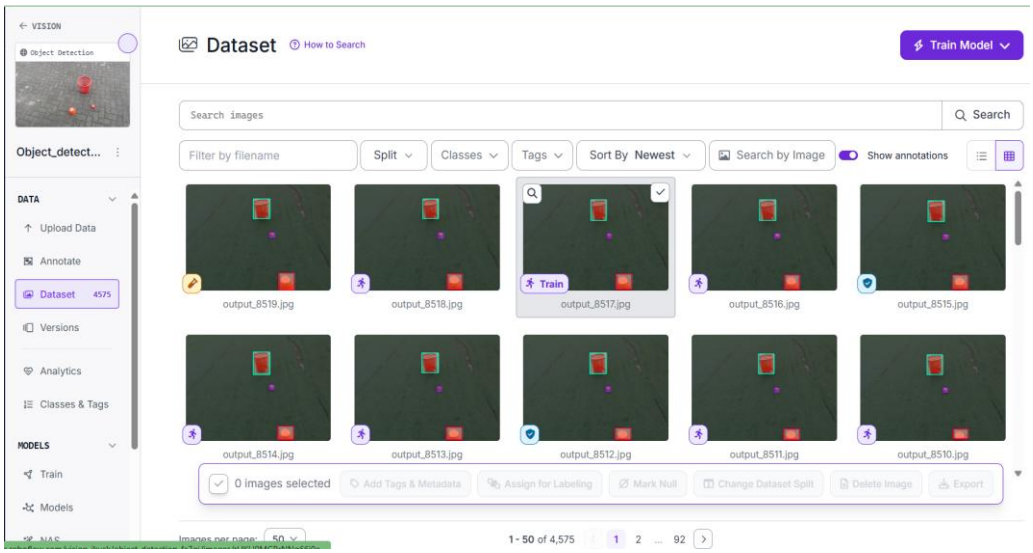

Figure 2. Example of labeled images in the Roboflow workspace.

2.2 Architecture Model and training process

The model used is YOLOv5n (nano), a lightweight variation of the YOLOv5 family with approximately 1.9 million parameters and a file size of around 3.8 megabytes, making it compatible with edge devices such as the Raspberry Pi 5. Its architecture consists of three prediction scales in the detector head (1/8, 1/16, and 1/32 of the input resolution) to modify objects of different sizes, feature extraction in the backbone, and a Path Aggregation Network in the neck for multi-scale fusion and Head [Figure3].

To enhance the model's generalization capabilities across varied environmental condition, multiple data augmentation strategies were employed during training this encompassed random scaling, horizontal flipping, and illumination modifications to replicate variances in object appearance and environmental conditions.

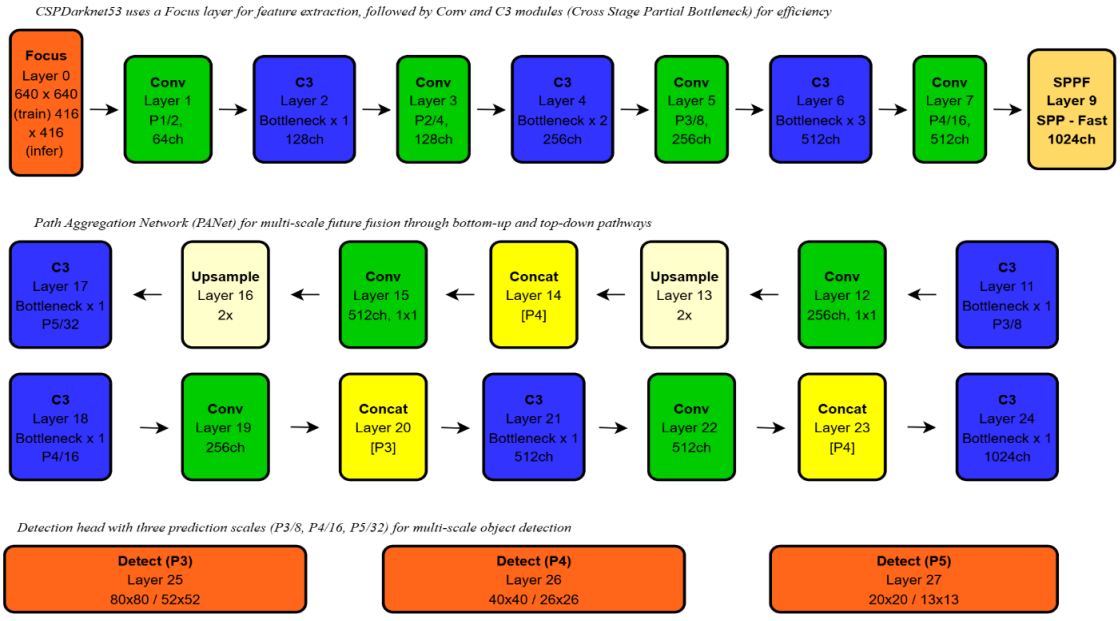

Figure 3. YOLOv5n Architecture.

In figure 3 CSPDarknet53 uses a Focus layer (6x6 Conv2d stride=2, which reduces the resolution to 320x320 pixels) to reduce the input image size from 640×640 pixels. This is followed by 4 C3 modules (Cross Stage Partial bottleneck for efficiency and parameter reduction) and SPPF (Spatial Pyramid Pooling Fast) for a receptive field that can detect objects of various sizes in a single forward pass. Path Aggregation Network (PANet) uses a top-down path (2x up sampling from P5 20×20 to P4 40×40, then to P3 80×80 via

skip concatenation connections) and a bottom-up path for P3/8, P4/16, and P5/32 feature fusion, ensuring information from various resolutions is available for accurate detection.

The YOLOv5 head generates a final prediction tensor using three grid scales: 80x80 (small objects such as distant cubes), 40x40 (medium objects such as normal toruses), and 20x20 (large objects such as nearby baskets). Each grid uses three box anchor points to generate box bounds (x,y,w,h), confidence scores, and probability classes (basket/cube/torus). The analysis process was performed on a workstation (Table 1) with the following parameters: 640×640 pixel input, batch size 16, approximately 150 epochs, SGD optimizer (standard momentum decay and YOLOv5 weights), CIoU loss (bounding box regression), BCE classification loss (3 classes), and object loss (object presence/absence). Each epoch was evaluated using mAP@0.5 on the validation set until best convergence.pt, then converted to OpenVINO IR format (.xml +.bin) for real-time inference on Raspberry Pi 5 (~15-20 FPS target).

2.3 Accuracy Evaluation

Accuracy, recall, F1-score, average precision (AP), and mean average precision (mAP) at several Intersection over Union (IoU) values are used to evaluate model accuracy. For each class, the number of true positives (TP), false positives (FP), and false negatives (FN) form the basis of this evaluation method. While recall measures the proportion of correctly detected objects relative to all ground-truth objects in the data, precision measures the proportion of correct detections relative to all predicted objects. In this context, TP, FP, and FN respectively represent correctly detected objects, false positive predictions, and missed detections. The precision and recall are defined mathematically as in (1) and (2):

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

The F1-score is defined as the harmonic mean between recall and precision, providing a single measure that balances sensitivity and detection reliability, as shown in (3):

$$F1 - score = 2x \frac{Precision\ x\ Recall}{Precision+Recall} \quad (3)$$

The Average Precision (AP) for each class is obtained from the area under the precision-recall curve at the current IoU threshold, while mAP is calculated as the average AP across each class for one or more IoU values. In addition, a confusion matrix is used to visualize the distribution of correct and incorrect classifications among classes so that classification problems can be analyzed more thoroughly; the corresponding numerical metrics table and confusion matrix are displayed and discussed in the results section. It is important to highlight that the metrics presented are based on a singular training-validation-test partition (Section 2.1); this study did not conduct repeated trials with alternative random splits or k-fold cross-validation, which is suggested as a future research avenue to validate the consistency of the findings.

2.4 Conversion to OpenVINO IR

The YOLOv5n model works optimally in Torch format (best.pt) before being converted to OpenVINO Intermediate Representation (IR) format, which consists of a model.xml file (network structure) and a model.bin file (trained weights). The model inference results produce normalized bounding box coordinates, object location scores, and class probabilities. This data is then further processed by normalizing the coordinates to the original image dimensions, applying Non-Maximum Suppression (NMS) with an IoU of 0.45, and filtering detections below a confidence threshold of 0.43. This on-device post-processing framework, where NMS and confidence filtering occur locally on the Raspberry Pi 5 instead of being transferred to an external host, was chosen to reduce communication latency and maintain a self-sufficient detection pipeline that does not rely on network connectivity, a crucial design aspect for an onboard VTOL navigation system. In comparison to the original PyTorch model, the OpenVINO IR conversion is anticipated to enhance CPU inference efficiency mainly via graph-level optimizations like operator fusion, constant folding, and memory-layout optimization for the ARM Cortex-A76 target; however, a direct and controlled latency comparison between the unconverted PyTorch model and the OpenVINO IR model on the identical Raspberry Pi 5 hardware was not conducted in this study, which is recognized as a valuable addition for future research to accurately measure this contribution.

2.5 Implementation on Raspberry Pi 5 and Real-Time Benchmarking

The inference phase is conducted solely on a Raspberry Pi 5 Rev 1.0 (quad-core ARM Cortex-A76 processor @2.4 GHz, 8 GB LPDDR4X RAM) utilizing a Logitech C920 USB camera (720p) and the

Raspberry Pi 64-bit operating system. The software ecosystem includes Python 3.11 alongside OpenVINO 2023.3, OpenCV, NumPy, and several auxiliary libraries for video processing, analysis, and visualization. The output of real-time detection is obtainable thru the host laptop without compromising the local inference capabilities of the device.

To ensure effective real-time performance on platforms with limited resources, various optimization measures have been implemented. The input resolution is set to 416×416 pixels to strike a balance between computational efficiency and detection accuracy. This decision illustrates a clear compromise: enhancing the input resolution to 640×640 pixels is anticipated to boost detection precision for diminutive or remote objects, albeit at the expense of increased inference latency and diminished FPS, while a reduced input size would expedite inference time but jeopardize the detection of small targets; consequently, the 416×416 configuration was chosen empirically as a pragmatic equilibrium for the object dimensions and distances (1–3 m) observed in this research. Additionally, non-essential background activities are minimized so that additional processing resources can be allocated to the inference pipeline.

The system operates in continuous frame processing mode, where each captured frame is processed instantly without temporary storage delays. This method minimizes the latency between image capture and identification results, which is critical for real-time vision-based applications such as autonomous navigation. OpenVINO enhances inference efficiency by optimizing the model for execution on ARM-based CPUs, thereby reducing inference time and improving overall system responsiveness. The foremost hardware limitation encountered during deployment was CPU-constrained inference on the Cortex-A76 cores, as the Raspberry Pi 5 lacks a dedicated neural accelerator; CPU usage during detection averaged approximately 81.7% (Section 3.3), resulting in restricted capacity for simultaneous onboard operations like flight-control calculations, a factor that must be taken into account when incorporating this pipeline into a comprehensive VTOL system.

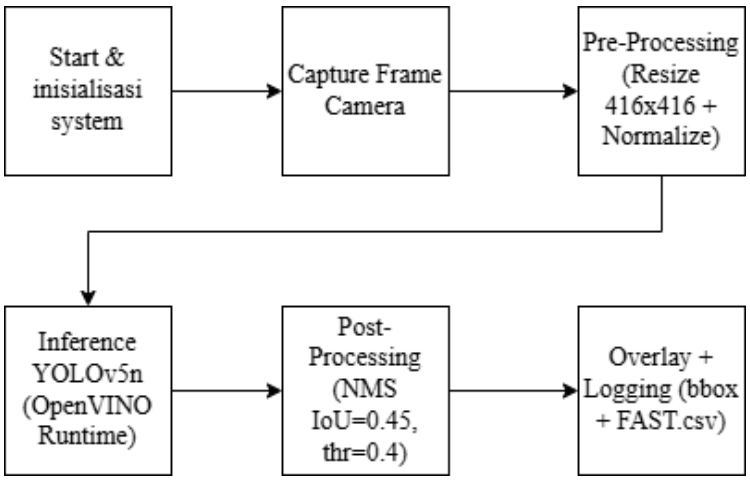

Figure 4. Complete Real-time Object Detection Pipeline using YOLOv5n-OpenVINO on Raspberry Pi 5.

In Figure 4, the detection pipeline continuously captures frames from the USB camera, performs pre-processing (resize to 416×416, normalization), executes YOLOv5n inference via OpenVINO Runtime, applies post-processing (NMS IoU 0.45), and logs metrics to benchmark_FAST.csv. A total of 2,160 frames were benchmarked to compute per-frame inference time (infer_ms), instantaneous FPS, CPU usage, and statistical summaries (mean, std) for performance characterization.

3. RESULTS AND DISCUSSION

Results from training (Section 2.2) and Raspberry Pi 5 demonstrate the effectiveness of the proposed pipeline.

3.1 Training Convergence Analysis

The training and validation losses for the regression, classification, and distribution focus loss (DFL) components are shown over 150 epochs in Figure 5. The validation loss closely follows the training curve without deviating, but all three loss terms decrease monotonically and converge well. With no clear indication of overfitting or underfitting during training, this behavior indicates that the model can successfully learn the underlying data distribution. The stable final loss value at a relatively low level provides a strong basis for achieving high detection accuracy in testing and implementation scenarios.

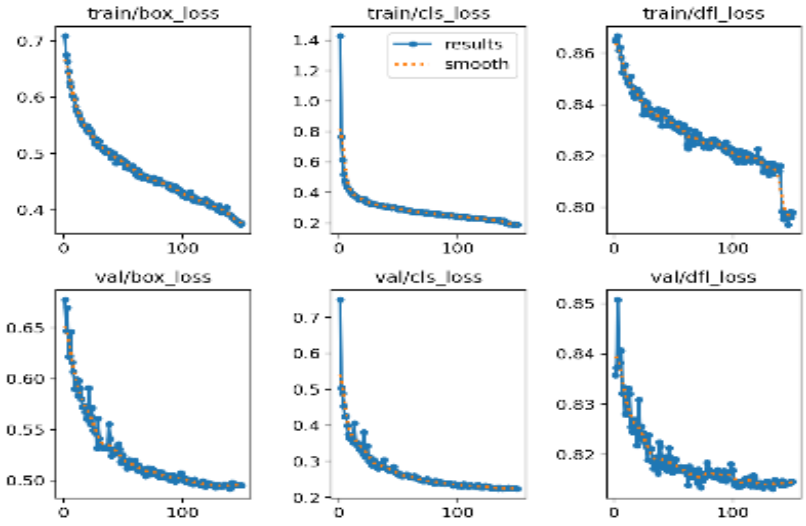

Figure 5. Training and Validation Loss Curves.

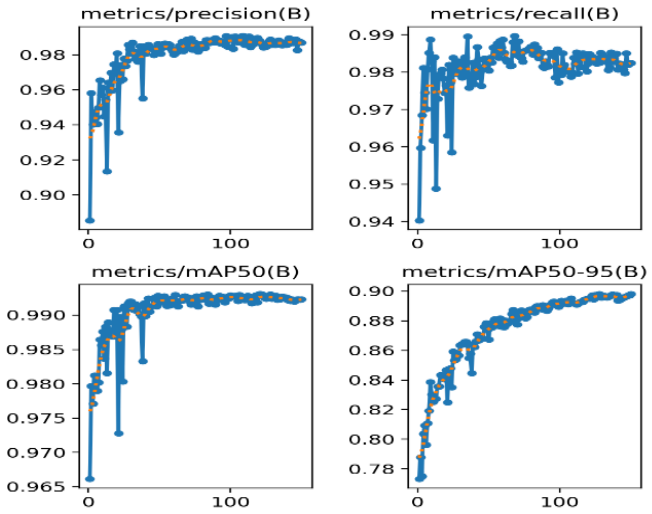

Figure 6. Precision, recall, mAP@0.5, and mAP@0.5:0.95.

Figure 6 shows the progress of the main evaluation metrics—mAP@0.5, and mAP@0.5:0.95—calculated on the validation set. In the early epochs, all metrics increased rapidly. As training progressed, these metrics eventually reached a saturation point and converged to stable values near the end of training. The model successfully learned highly discriminative features for the three object classes (baskets, cubes, and tori), as indicated by stable precision and recall at 0.98, mAP@0.5 reaching around 0.99, and mAP@0.5:0.95 reaching around 0.87 in the last epoch. The final YOLOv5n checkpoint was selected for deployment on the Raspberry Pi 5 in subsequent real-time experiments based on these results.

3.2. Classification accuracy results

Table 2. Performance Metrics

Metric	Bucket	Cube	Torus	Average
Precision	0.994	0.989	0.994	0.992
Recall	0.994	0.989	0.994	0.992
F1-score	0.994	0.989	0.994	0.992
mAP@0.5	0.994	0.989	0.994	0.992
Support	673	689	680	2,042

A summary of the metric calculations is shown in Table 2, where the average mAP@0.5 is approximately 0.992 and the F1-score is approximately 0.992 at a confidence threshold of 0.43, indicating that the model can detect and localize objects with very high accuracy for the three classes.

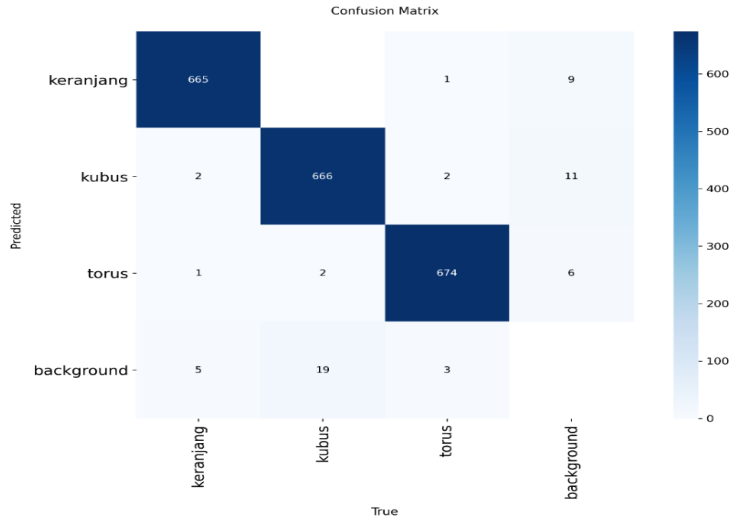

Figure 7. Confusion Matrix.

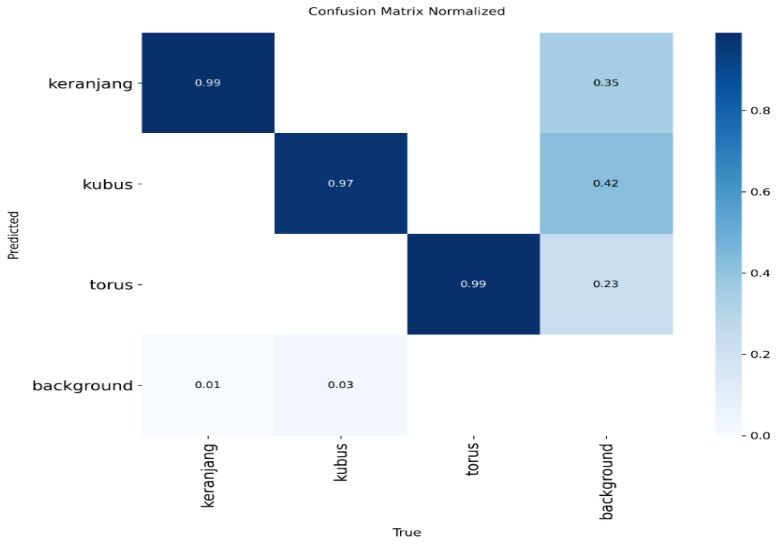

Figure 8. Confusion Matrix Normalized.

In addition, a confusion matrix is used to illustrate correct and incorrect predictions for each class so that classification error patterns can be analyzed more closely, as shown in Figures 7 and 8. The confusion matrix in Figure 7 illustrates that the basket, cube, and torus classes consistently have 665 out of 673, 666 out of 689, and 674 out of 680 examples detected correctly, while the rest are misclassified between classes or undetected (background). The resulting accuracy values are approximately 99.0% for baskets, 96.7% for cubes, and 99.1% for tori per class. Figure 8 shows the normalized confusion matrix, where the diagonal elements for the first three classes are between 0.97 and 0.99, while the diagonal elements are below 0.01 to 0.04. This indicates that the model has excellent ability to distinguish classes with a very small number of misclassifications.

3.3 Real-Time Outdoor Performance

Real-time testing in an outdoor environment using Raspberry Pi 5 shows that the YOLOv5n-based object detection system optimized with OpenVINO can consistently detect three types of objects in various environmental conditions and everyday situations. Frame rates during field testing were around 17–19 frames per second with an inference time per frame of approximately 52–58 ms, meeting the real-time requirements for vision-based autonomous navigation.

Figure 9 shows the experimental setup of the VTOL platform during real-time testing, where the drone operates under controlled tethered conditions to ensure stability and safety. This setup enables evaluation

under semi-realistic conditions, where onboard computation and environmental variability affect detection performance.

Figure 9. VTOL drone platform equipped with Raspberry Pi 5 during real-time outdoor testing under controlled tethered conditions.

Based on the experimental setup shown in Figure 9, the system was evaluated under varying object distances and environmental condition. Within a range of 1 to 3 meters, three objects can be inspected simultaneously with confidence scores generally ranging from 0.70 to 0.80. However, in less than ideal conditions or with objects still in use, the lowest confidence score remains around 0.70. This shows that the model remains stable in handling variations in object position and orientation during testing. Additionally, the system did not show a significant decline in detection performance when the background changed (e.g., grass, or concrete areas), indicating that the model has the ability to generalize to various outdoor scenarios. Nonetheless, the outdoor assessment detailed herein was performed under moderate circumstances, and it is anticipated that detection efficacy would diminish in more arduous situations not explicitly investigated in this research, including significant object obstruction, rapid relative movement between the drone and target objects resulting in motion blur, or densely cluttered outdoor environments; a systematic evaluation of the system under these conditions is reserved for subsequent studies.

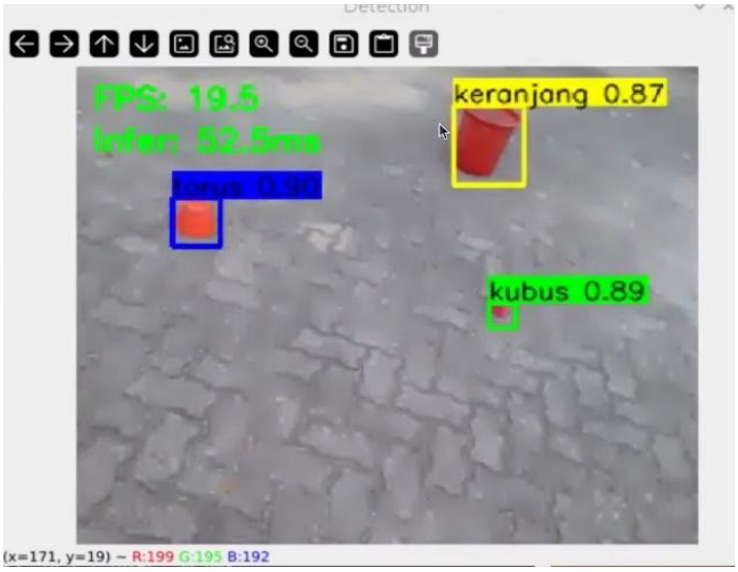

Figure 10. Outdoor detection result (FPS=19.5, inference=52.5ms).

In Figure 10, this result illustrates detection in an outdoor environment with paving, where three objects are detected with a confidence level of at least 0.87, even when there is variation in the background texture. The overlay shows an FPS of around 19.5 and an inference time of 52.5 ms, confirming that the system is responsive in external environments with more complex surfaces. High confidence scores (>0.87) across various outdoor backgrounds—from textured paving and natural grass—demonstrate the model's strong generalization ability after being tested on a 3-class dataset (about 3,420 labeled examples for each category).

This robustness is crucial for autonomous navigation applications, where environmental conditions are unpredictable. To validate the Non-Maximum Suppression (NMS IoU=0.45) configuration, 3-object detection that consistently produces minimal false positives must be examined. This will ensure that bounding boxes remain stable even when objects are in close proximity to the camera. These field results, when combined with a validation score of 0.992 on mAP@0.5, demonstrate strong generalization across diverse outdoor conditions.

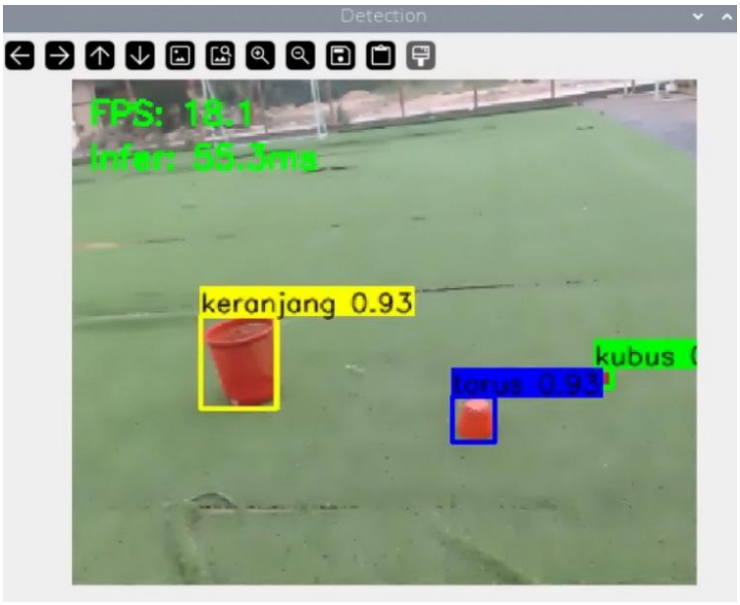

Figure 11. Outdoor detection result (FPS=18.1, inference=55.3 ms).

In Figure 11, the three objects are closer to the camera, so the relative size of the bounding boxes is smaller. However, detection remains accurate with a confidence level of around 0.93 for the basket and torus. The FPS, which is around 18.1 with an inference time of 55.3 ms, indicates that distance variations do not significantly affect system performance when objects are still within the effective resolution range of the model.

Table 3. Sample Real-Time Performance Log (selected frames: 3 objects detected, FPS ≥17)

	Infer (ms)	FPS	CPU (%)	Num Detections
	40.68	23.18	80.80	3
	43.00	21.74	78.60	3
	41.97	21.69	81.50	3
	56.89	16.39	83.30	3
	40.80	22.80	80.80	3
	45.87	20.35	81.50	3
	47.54	18.41	87.90	3
	53.47	17.39	83.30	3
	34.63	26.17	80.00	3

	35.67	24.75	80.10	3
Average	42.59	21.76	81.70	3

Table 3 presents a sample of the real-time performance log on the Raspberry Pi 5, specifically for frames where all three target objects are detected simultaneously and the system operates at 17 FPS or higher. Under these optimal detection conditions, the average inference time is approximately 42.6 ms with an average throughput of about 21.8 FPS and CPU utilization around 81.7%. It is important to note that the overall outdoor benchmark across all 2,160 frames — encompassing varying object distances, counts, and environmental conditions — yields an average inference time of approximately 55 ms and throughput of approximately 18 FPS, as reported in Section 3.3. These results confirm that the OpenVINO-optimized YOLOv5n model is capable of maintaining real-time multi-object detection under the evaluated outdoor conditions.

Figure 12 illustrates the identical ten sampled frames detailed in Table 3, illustrating the correlation between per-frame inference duration and frames per second (FPS). The data illustrates a negative association between the two variables: frames exhibiting greater inference delay, such F4 at 56.89 ms, are associated with diminished FPS (16.39), whereas frames with reduced latency, such as F9 at 34.63 ms, are linked to elevated FPS (26.17). This verifies that inference duration is the primary limitation on real-time throughput on the Raspberry Pi 5. In practical applications, an inference duration between 35 and 55 milliseconds and a throughput of about 18 to 22 frames per second are adequate for real-time visual feedback in VTOL navigation activities like target tracking and obstacle evasion, with update frequencies exceeding roughly 10 Hz typically deemed sufficient for stable closed-loop regulation.

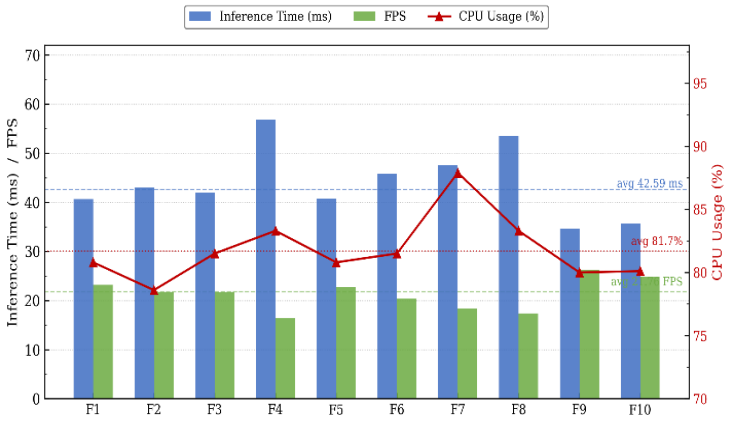

Figure 12. Real-time performance analysis of YOLOv5n with OpenVINO on Raspberry Pi 5 across 10 sampled frames.

4. CONCLUSION

Using YOLOv5n with OpenVINO optimization on a Raspberry Pi 5, this study successfully demonstrated a full three-class object detection pipeline. It achieved a mAP@0.5 of 0.992, an F1-score of 0.992 at a confidence threshold of 0.43, an average inference time of about 55 ms per frame, and a real-time throughput of about 18 FPS across a 2,160-frame outdoor benchmark with CPU utilization of 81.70%. With per-class precision and recall both reaching 0.992, the model showed dependable detection of basket, cube, and torus objects under a variety of outdoor conditions, including different illumination levels, object-to-camera distances of 1-3 meters, and heterogeneous surface backgrounds like grass, and concrete. Confusion matrix analysis validated the robustness of the learnt feature representations by confirming that the model did not cause any cross-class misclassification. A crucial optimization step that allowed the ARM Cortex-A76 CPU to maintain real-time inference without the need for a specialized hardware accelerator was the conversion from PyTorch to OpenVINO IR format at 416x416 pixels. The proposed pipeline is a lightweight and efficient solution for vision-based object recognition in VTOL autonomous navigation applications with limited resources. Despite these results, a number of limitations should be noted, such as the limited coverage of the dataset, the tethered low-altitude test settings, and the lack of temporal tracking for objects moving quickly.

These constraints suggest that the documented performance ought to be viewed as indicative of regulated, tethered, low-altitude flight in moderate outdoor environments, rather than as a reflection of completely autonomous free flight; the efficacy in untethered flight dynamics, elevated altitudes, or complex obstacles requires validation in subsequent research. These restrictions will be addressed by future research in a number of focused ways. To support complete VTOL autonomous situations, the dataset will be extended to include obstacle and landing pad classes. To further lower inference latency and maybe surpass 24 FPS throughput, INT8 post-training quantization will be investigated. To enable closed-loop, vision-

guided autonomous landing, the detecting system will be connected to a PX4 autopilot controller using the MAVLink protocol. To enhance temporal consistency between frames, multi-object tracking methods like Deep SORT will be used. Additionally, potential improvements in inference speed and energy efficiency for extended autonomous flights will be assessed by evaluating hardware acceleration alternatives such as Google Coral TPU and Intel Neural Compute Stick 2 (NCS2).

Future research will also assess prolonged operation under continuous high computational demand, including potential thermal throttling on the Raspberry Pi 5 and its impact on long-term inference stability. Furthermore, while the existing framework has been validated on three object classes pertinent to this VTOL application, its modular training–optimization–deployment pipeline is not intrinsically confined to these classes and could be modified for supplementary object categories or other edge-vision tasks by retraining the YOLOv5n model on a newly annotated dataset, without altering the OpenVINO deployment architecture.

ACKNOWLEDGMENTS

This research was supported by Batam State Polytechnic.

FUNDING INFORMATION

This study was not supported by external funding.

AUTHOR CONTRIBUTIONS STATEMENT

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ryan Satria Wijaya	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Astuti Yeirene	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors assert no conflict of interest for the publication of this research.

INFORMED CONSENT

Not relevant. This research did not include human subjects, personal information, or recognizable photos of individuals; hence, informed permission was unnecessary.

ETHICAL APPROVAL

This research did not necessitate ethical approval.

DATA AVAILABILITY

The data underpinning the findings of this investigation can be obtained from the corresponding author upon a reasonable request. Although the complete dataset is currently not available to the public, the trained model weights, OpenVINO IR files, and real-time benchmarking scripts can be provided by the associated author upon a reasonable request to provide independent validation of the given results.

REFERENCES

[1] T. G. Tang, J. Ni, Y. Zhao, Y. Gu, and W. Cao, "A Survey of Object Detection for UAVs Based on Deep Learning," *Remote Sens.*, vol. 16, no. 1, pp. 1–29, 2024, doi: 10.3390/rs16010149.

[2] S. Kang, Z. Hu, L. Liu, K. Zhang, and Z. Cao, "Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis," *Electron.*, vol. 14, no. 6, pp. 1–36, 2025, doi: 10.3390/electronics14061104.

[3] H. Wang, Z. Yang, Q. Liu, Q. Zhang, and H. Wang, "TCE-YOLOv5: Lightweight Automatic Driving Object Detection Algorithm Based on YOLOv5," *Appl. Sci.*, vol. 15, no. 11, pp. 1–19, 2025, doi: 10.3390/app15116018.

[4] R. Yu, Y. Zhang, and S. Liu, "CIMB-YOLOv8: A Lightweight Remote Sensing Object Detection Network Based on Contextual Information and Multiple Branches," *Electron.*, vol. 14, no. 13, 2025, doi: 10.3390/electronics14132657.

[5] A. V. R. Katkuri, H. Madan, N. Khatri, A. S. H. Abdul-Qawy, and K. S. Patnaik, "Autonomous UAV navigation using deep learning-based computer vision frameworks: A systematic literature review," *Array*, vol. 23, no. March, p. 100361, 2024, doi: 10.1016/j.array.2024.100361.

[6] J. Xiao, R. Zhang, Y. Zhang, and M. Feroskhan, "Vision-Based Learning for Drones: A Survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 36, no. 9, pp. 15601–15621, 2025, doi: 10.1109/TNNLS.2025.3564184.

[7] P. Mittal, *A comprehensive survey of deep learning-based lightweight object detection models for edge devices*, vol. 57, no. 9. Springer Netherlands, 2024. doi: 10.1007/s10462-024-10877-1.

[8] Raspberry Pi Foundation, "Raspberry Pi 5 product brief." [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/.
[9] W. Kong, D. Zhou, D. Zhang, and J. Zhang, "Vision-based autonomous landing system for unmanned aerial vehicle: A survey," in *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, 2014, pp. 1–8. doi: 10.1109/MFI.2014.6997750.
[10] D. Minott, S. Siddiqui, and R. J. Haddad, "Benchmarking Edge AI Platforms: Performance Analysis of NVIDIA Jetson and Raspberry Pi 5 with Coral TPU," in *SoutheastCon 2025*, 2025, pp. 1384–1389. doi: 10.1109/SoutheastCon56624.2025.10971592.
[11] D. Ngo, H. C. Park, and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments," *Electronics (Switzerland)*, vol. 14, no. 12, pp. 1–54, 2025. doi: 10.3390/electronics14122495.
[12] H. Ahn *et al.*, "Performance Characterization of Using Quantization for DNN Inference on Edge Devices," *Proc. - 7th IEEE Int. Conf. Fog Edge Comput. ICFEC 2023*, pp. 1– 6, 2023, doi: 10.1109/ICFEC57925.2023.00009.
[13] A. Swaroop, A. Satsangi, M. Sameer, and G. Ahmad, "Performance Evaluation of YOLOv5 and YOLOv8 for Vehicle Detection: A Comparative Study," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10723901.
[14] T. Diwan, G. Anirudh, and J. V Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, 2023, doi: 10.1007/s11042-022-13644-y.
[15] Q. L. Trieu, B. Javadi, J. Basilakis, and A. N. Toosi, "Performance Evaluation of Serverless Edge Computing for Machine Learning Applications," *Proc. - 2022 IEEE/ACM 15th Int. Conf. Util. Cloud Comput. UCC 2022*, pp. 139–144, 2022, doi: 10.1109/UCC56403.2022.00025.
[16] N. Surantha and N. Sutisna, "Key Considerations for Real-Time Object Recognition on Edge Computing Devices," *Applied Sciences (Switzerland)*, vol. 15, no. 13, pp. 1–26, 2025. doi: 10.3390/app15137533.
[17] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," in *Low-Power Computer Vision*, 2022, pp. 291–326. doi: 10.1201/9781003162810-13.
[18] Z. Ren, H. Zhang, and Z. Li, "Improved YOLOv5 Network for Real-Time Object Detection in Vehicle-Mounted Camera Capture Scenarios," *Sensors*, vol. 23, no. 10, 2023, doi: 10.3390/s23104589.
[19] E. Y. Pradana, S. A. Aji, M. A. Abdulrrozaq, A. H. Alasiry, A. Risnumawan, and E. Pitowarno, "Optimizing YOLOv8 for Real-Time Performance in Humanoid Soccer Robots with OpenVINO," in *2024 International Electronics Symposium (IES)*, 2024, pp. 304–309. doi: 10.1109/IES63037.2024.10665829.
[20] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. Da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electron.*, vol. 10, no. 3, pp. 1–28, 2021, doi: 10.3390/electronics10030279.
[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, 2016, doi: 10.1109/TPAMI.2015.2437384.
[22] H. Shakhatreh *et al.*, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019, doi: 10.1109/ACCESS.2019.2909530.
[23] A. Alotaibi, H. Alatawi, A. Binnouh, L. Duwayriat, T. Alhmiedat, and O. M. Alia, "Deep Learning-Based Vision Systems for Robot Semantic Navigation: An Experimental Study," *Technologies*, vol. 12, no. 9, pp. 1–19, 2024, doi: 10.3390/technologies12090157.
[24] L. Rey *et al.*, "A Performance Analysis of You Only Look Once Models for Deployment on Constrained Computational Edge Devices in Drone Applications," *Electron.*, vol. 14, no. 3, pp. 1–24, 2025, doi: 10.3390/electronics14030638.
[25] D. Cantero, I. Esnaola-Gonzalez, J. Miguel-Alonso, and E. Jauregi, "Benchmarking Object Detection Deep Learning Models in Embedded Devices," *Sensors*, vol. 22, no. 11, pp. 1–25, 2022, doi: 10.3390/s22114205.
[26] Y. Chang, Y. Cheng, U. Manzoor, and J. Murray, "A review of UAV autonomous navigation in GPS-denied environments," *Rob. Auton. Syst.*, vol. 170, no. September 2022, p. 104533, 2023, doi: 10.1016/j.robot.2023.104533.
[27] Z. Li, H. Li, and L. Meng, "Model Compression for Deep Neural Networks: A Survey," *Computers*, vol. 12, no. 3, pp. 1–22, 2023, doi: 10.3390/computers12030060.

BIOGRAPHIES OF AUTHORS

Ryan Satria Wijaya Assistant Professor, a lecturer at Politeknik Negeri Batam specializing in Robotics, Motion Control, Computer Vision, and Artificial Intelligence, has contributed to various research projects and publications in these fields. His work includes studies on deep learning algorithms for object detection, path planning in service robots, and the development of interactive learning media for vocational education. His research aims to advance technological applications in robotics and AI, enhancing both academic understanding and practical implementations. He can be contacted at email: ryan@polibatam.ac.id.

Astuti Yeirene Panggabean is a fresh graduate in the Robotic Engineering study program, Department of Electrical Engineering, Politeknik Negeri Batam. Her research interests encompass computer vision, real-time object detection, embedded artificial intelligence systems, and edge computing implementation on resource-constrained platforms such as the Raspberry Pi and dataset preparation and annotation, model training and performance evaluation, OpenVINO optimization, and real-time benchmarking of the YOLOv5n object detection system deployed on a Raspberry Pi 5 for vertical take-off and landing (VTOL) drone applications. She can be contacted at email: panggabeanastutiyeirene@gmail.com.