บทความวิจัย (Research Article)

Performance Analysis of Low-Cost Wireless AIoT for Real-Time Object Detection: Arowana Monitoring Case Study

Wiklom Teerapabakjorndet* and Nonthawat Somtad

Department of Electrical and Biomedical Engineering, Faculty of Engineering, Prince of Songkla University
* Corresponding author
wiklom.t@psu.ac.th

Received: 3 Dec 2024
Revised: 31 Mar 2025
Accepted: 2 Apr 2025

Abstract

This study presents the implementation of cost-efficient Artificial Intelligence of Things (AIoT) systems and evaluates the impact of wireless IoT networks on real-time performance of object detection rates. The detection rates considered in this paper incorporate both inference rates from AI computations and image frame arrival rates from IoT networks. As a practical application, the case study examines arowana monitoring systems in aquarium environments. The research utilizes the ESP32-CAM for video capture and the YOLOv3-tiny model for object detection, with HTTP over Wi-Fi serving as the communication protocol for the deployed wireless IoT networks. Numerical analyses demonstrate that the object detection rate is improved when the network goodput for a given image size allows the frame arrival rate to match the inference rate. However, when the goodput results in an arrival rate exceeding the inference rate, further improvements in detection rates are not observed. Conversely, insufficient goodput leading to arrival rates below the inference rate results in detection rates aligning with the frame arrival rate. The network goodput performance is shaped by wireless IoT mechanisms that address contention, collision, congestion, and error control. Consequently, effective integration of wireless network functionality with object detection systems is essential to meet the stringent requirements of efficiency, reliability, and timeliness in cost-efficient real-time AIoT systems. This underscores the importance of harmonizing wireless IoT networks and object detection systems to enhance the performance and deployment of AIoT systems in real-world applications.

Keywords: AIoT, ESP32, YOLOv3-tiny, Network Optimization, Aquaculture

1. Introduction

The rapid advancements in AI technologies, coupled with the decreasing costs of IoT devices, have paved the way for the integration of these technologies into the emerging field of real-time AIoT [1] [2]. This paper presents the progress of

our ongoing research project, which investigates the practical and efficient application of IoT and AI in the aquaculture of the Asian arowana within aquarium tanks. The care of this species is challenging due to its high susceptibility to stress from various factors [3]. Consequently, our research is driven by the need to assist caregivers in promptly detecting and addressing the health issues of the arowana.

Several research efforts have been documented in the literature regarding the development of IoT systems for the Asian arowana [4] [5]. These studies primarily focus on basic IoT systems designed for monitoring and controlling aquarium tanks. However, the application of IoT and AI to diagnose illnesses in the arowana is not studied. Various symptoms can be visibly detected on the arowana's body. Additionally, abnormal behaviors, which may occur in the absence of visible signs, can be observed by caregivers. These behaviors include remaining still at the bottom of the aquarium and swimming more slowly. It is impractical for caregivers to continuously monitor the arowana for these symptoms. Consequently, our long-term research aims to explore the integration of IoT and AI technologies for the aquaculture of the Asian arowana in aquarium tanks, focusing on monitoring its well-being through external appearance and behavior. Specifically, our research utilizes a camera to capture video images of the arowana, which are then wirelessly transmitted over IoT networks to an edge device for AI processing.

To identify the research gap in the engineering domain, we review literature from other applications, even though they are not specifically developed for arowana aquaculture. In [6], IoT and AI systems for monitoring ornamental fish in aquarium tanks are developed. These systems use a Raspberry Pi with a camera to count fish, while a smartphone is employed to photograph fish for illness detection from their external appearances via a web server. Although similar to our research goals, the camera in [6] must be closely connected to the edge device, and fish illness is not examined locally. In [7], an IoT prototype for monitoring and controlling seahorse aquaculture in an aquarium is developed, with an AI edge computing device based on YOLOv3-tiny algorithms used near the aquarium tank to detect seahorse illness from camera images. This work is somewhat aligned with our research project but does not consider video streaming for irregular behavior detection.

Our research investigates the impact of wireless IoT networks on the object detection rates of video frames in AIoT systems. The study aims to address an existing research gap and establish a foundation for designing practical and cost-effective AIoT systems. To demonstrate feasibility, we selected a use case involving the monitoring of arowanas on a large aquaculture farm, requiring numerous aquarium tanks to be individually observed. Each tank necessitates at least one IoT camera for close monitoring, resulting in a high demand for affordable camera devices. To address this, the ESP32-CAM is chosen as the IoT camera, coupled with the YOLOv3-tiny model to enable low-computation object detection.

This research paper primarily explores simpler AIoT systems, specifically those with a single camera device monitoring one aquarium tank. The pretrained YOLOv3-tiny model is deployed on a laptop, serving as an edge device for object detection during experiments. By focusing on a simplified setup, this study provides a foundation for future research on larger, more complex real-time object detection systems. From a technical perspective, the study analyzes the end-to-end latency associated with consecutive image frames in real-time

streaming video between wireless IoT networks and AI edge computing devices in prototype systems. To integrate multidisciplinary knowledge about the effects of wireless IoT networks on object detection rates for real-time AIoT, this research emphasizes a fundamental understanding of these interactions.

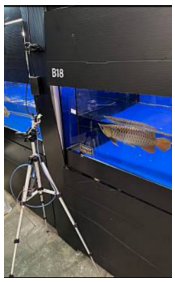
The structure of this paper is organized as follows. Firstly, the systems and models deployed in this work are described. Next, the experimental design used to evaluate our prototypes is outlined. Then, the numerical results and discussions are given. Finally, the conclusions of this paper are presented.

2. Systems and Models

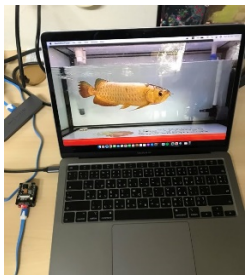
This section describes the wireless object detection systems deployed for arowana monitoring. These systems, illustrated in Figure 1, comprise four main components: ESP32-CAM, ESP32 access point (AP), a standard laptop, and YOLOv3-tiny object detection. The details of these systems and models are elaborated in this section.

The block diagram of the deployed systems is illustrated in Figure 2. The wireless IoT networks utilize an HTTP client-server architecture, where the ESP32-CAM functions as the client and the laptop serves as the server. An intermediate node, configured as a Wi-Fi access point (AP), facilitates this end-to-end wireless communication. This intermediate node is another ESP32 operating in Wi-Fi AP mode.

The performance analysis of multi-hop Wi-Fi networks is well-documented in the literature [8] [9] [10] [11]. Building on this foundation, our study specifically examines the interplay between multi-hop IoT networks and real-time AI object detection systems. This approach shifts the focus from traditional network performance analysis to exploring how these networks impact the performance of AIoT.



a) Wireless ESP32-CAM



b) ESP32 access point

c) YOLOv3-tiny object detection

Figure 1 The deployed wireless object detection systems for arowana monitoring

This study implements a two-hop wireless communication system to gain initial insights before assessing the impact of additional hops in future research. Upon receiving an image, the HTTP server on the laptop stores it in MySQL database before processing it with YOLOv3-tiny object detection algorithms. For the remainder of this paper, this laptop will occasionally be referred to as the laptop server.

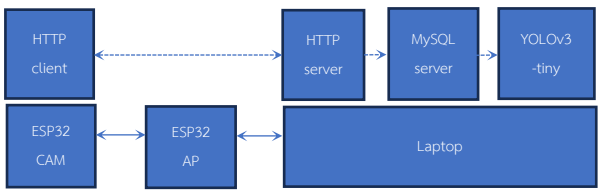


Figure 2 System architecture

In summary, the system operations can be divided into three main procedures: 1) Transmitting an

image wirelessly from the ESP32-CAM client to the laptop server. 2) Storing each image in the MySQL database. 3) Retrieving an image from the MySQL database and detecting the arowana within it. These three procedures can operate concurrently. Next subsections detail important components of our systems and models employed in this work.

2.1 ESP32 Software Components

In this work, the ESP-IDF software libraries [12] are employed to develop wireless communication protocols on ESP32 nodes. Specifically, the required protocols based on the TCP/IP layering model include HTTP, TCP, IP, and Wi-Fi. The source codes for these protocols are organized into distinct components within ESP-IDF and can be easily integrated into ESP-IDF projects for specific applications. Additionally, Espressif's ESP-IDF GitHub provides several practical application examples of ESP-IDF projects.

This study initially utilizes and modifies the esp_http_client libraries from the protocol examples in ESP-IDF projects to meet the application requirement of transmitting arowana images from the ESP32-CAM to the database server. The esp_http_client example integrates multiple protocol components, enabling the ESP-IDF project to offer HTTP services over TCP/IP network stack and Wi-Fi connections. HTTP, an application layer protocol within the TCP/IP stack, comprises three primary steps: connection establishment, data exchange, and connection termination. In this study, the ESP32-CAM functions as the HTTP client, while the HTTP server is a laptop, which hosts a database server as well. The client initiates the HTTP connection to the server when it is prepared to transmit data, specifically when the ESP32-CAM captures an image of an arowana. Upon successful connection establishment, the ESP32-CAM transmits image data to the database server. Once the image transmission is complete, the HTTP connection between the client and server is terminated. The ESP32-CAM then proceeds to capture the next image of the arowana, and the HTTP communication process recommences. This cycle repeats continuously, with the database server storing all image frames for subsequent processing by the YOLOv3-tiny model deployed on the same laptop. The ESP-IDF protocol component utilized in this work to implement the HTTP client functionality on the ESP32-CAM is esp_http_client, which is based on HTTP version 1.1 (HTTP/1.1).

2.2 MySQL

The database is deployed using MySQL on the XAMPP platform. XAMPP, an open-source software suite, enables developers to easily build and test web services in local environments before deploying them to production on the Internet. XAMPP includes MySQL, which is essential for developing the local database server on our laptop. Additionally, XAMPP provides a convenient software environment for developers to create databases and tables using the database administration tool phpMyAdmin.

2.3 HTTP Server

This work implements the HTTP server using Werkzeug, a Python library within the Flask framework, on a Windows-based laptop. The HTTP server manages the process by which the ESP32-CAM client posts images to the database. Essentially, the laptop HTTP server processes HTTP requests initiated by the ESP32-CAM HTTP client. This study utilizes the POST method for HTTP requests. An HTTP POST request message comprises a request line, headers, and a body. The

body contains the content that the HTTP POST message intends to send to the server. In this work, the content is image data encoded in base64 format. Base64 is an encoding scheme that converts the binary data of each image into a sequence of ASCII characters, as defined in RFC 4648. Base64 is widely used for transmitting images via HTTP/1.1, a text-based protocol at the application layer employed in this study.

2.4 YOLOv3-tiny

YOLOv3 is the earliest YOLO model contributed and maintained by Ultralytics as open source, bringing significant improvements in usability and widespread adoption in research literature. While newer versions of YOLO offer advancements in accuracy and speed, YOLOv3 remains a robust choice for this study due to its maturity, strong community support, and suitability for our experimental setup. More specifically, YOLOv3-tiny was selected for this study due to its compact model size and high inference rate, which make it ideal for deployment on edge computing devices requiring efficient real-time processing and low latency. Future work will explore newer YOLO versions, as they hold potential for further improvements in accuracy and speed. These considerations highlight why YOLOv3-tiny was selected for our study, while also paving the way for future exploration of newer YOLO models.

This work utilizes the default network structure configuration of YOLOv3-tiny [13] [14]. The deployed YOLOv3-tiny resizes all input images to 416 x 416 pixels while maintaining their aspect ratios. Each image is scaled using an interpolation method to ensure its largest dimension matches the target size, though some image content may be lost in this process. To obtain the YOLOv3-tiny model for arowana detection, a dataset of arowanas in a fish tank was created. Video clips were recorded from arowana farms in Songkhla city, featuring healthy arowanas. The ESP32-CAM was used for video recording at three resolutions: QVGA, VGA, and SXGA. This initial study focuses on the effects of wireless IoT networks on AI object detection rates; hence, a dataset of sick arowanas was not created. Identifying abnormal behaviors requires specialized veterinary expertise, which we plan to address through future research collaborations with experts in this field.

The arowana dataset was created using the Roboflow service, a web-based tool for generating image datasets. Colab (Colaboratory), a Google cloud-based platform, was used to train the YOLOv3-tiny model for arowana detection. Two split datasets were used for training and validating the arowana detection model. Command line interface (CLI) and Python scripts were used to request services from the WandB (Weights & Biases) API to evaluate training results across various metrics. Training began with a pretrained yolov3-tiny.pt model, using the VGA resolution arowana dataset for training and validation. mAP (mean Average Precision) metrics are used to measure the accuracy of the YOLOv3-tiny model after training in this work. The mAP metrics provide a robust measure of detection accuracy based on the precision and recall curve at a given IoU threshold. Specifically, mAP50 evaluates detection accuracy at an IoU threshold of 50%, while mAP50-95 averages the results across multiple IoU thresholds for a comprehensive assessment. In the validation of our trained model, the mAP50 and mAP50-95 metrics achieved values of 0.994 and 0.895, respectively, demonstrating the high accuracy of the YOLOv3-tiny model in detecting Asian arowana.

This study uses PyTorch, a popular deep learning library, for deploying the YOLOv3-tiny model on our laptop. The trained weight file from the Colab training

session is stored locally on the laptop, along with the model configurations. The PyTorch-based detection model is instantiated from these weights and configurations and applied to each image retrieved from the MySQL database for arowana detection. The laptop used in this study for the MySQL database server and the pretrained YOLOv3-tiny model is an ASUS TUF Gaming F17 FX706HC, featuring an Intel Core i5-11400H CPU Processor (2.7 GHz, 6 Cores) with 8 GB RAM and IEEE 802.11b/g/n/ac/ax Wi-Fi. The operating system is Windows 11. Although the laptop includes a GPU, only the CPU was used for arowana detection in this study.

3. Experimental design

To ensure a controlled environment and avoid any disruptions at the arowana farm, the experiments were conducted at our department. The ESP32-CAM was configured to capture images of an arowana from a video played on a laptop, which served as a representation of a fish tank with an arowana. This video was recorded at SXGA resolution at an arowana farm in Songkhla city, as illustrated in Figure 1. The distances between the ESP32-CAM and the ESP32 AP were set to two ranges: 1 meter and 50 meters. There was a line of sight (LOS) for wireless communication between the ESP32-CAM and the ESP32 AP. Both devices were placed in the hallway with LOS, in our department. The ESP32 AP and the laptop server were placed close to each other. The 50-meter setup approaches the Wi-Fi coverage boundary, where the RSSI can occasionally drop below the acceptable level as specified in [15]. These two distance ranges are referred to as 1-meter and 50-meter wireless distances in our numerical analysis and discussion to avoid ambiguity in their meaning. The objective of setting these distinct ranges is to evaluate the different wireless effects on AIoT systems when (1) all participating nodes are within the closed proximity area and (2) the source node is at the Wi-Fi coverage boundary of the access point. These two extreme wireless scenarios demonstrate the interplay between IoT networks and real-time object detection systems.

To investigate the effects of image sizes on the system, the ESP32-CAM was configured to capture images at three different resolutions: QVGA, VGA, and SXGA. In all experiments, the ESP32-CAM is configured to capture images at a frame rate of 10 fps. Each experiment was replicated three times, with ten transmission images successfully received by the laptop in each replication. The average numerical results, along with their 95% confidence intervals, are presented. Five response variables were measured in these experiments.

1) Object detection rate: This metric, measured in frames per second (fps), quantifies the rate at which AIoT systems transmit and process object detection tasks. It is influenced by the hardware and software infrastructure of the computing unit, as well as the communication networks, including the laptop server, the YOLOv3-tiny model used in this experiment, and the wireless IoT networks. Let Υ_i^{AIoT} denote the object detection rate of AIoT systems for each image frame *i*. It is calculated using Equation (1), where t_i^d represents the time after completing the inference of the image frame *i*. The term $\left(t_i^d - t_{i-1}^d\right)$ reflects the detection time, encompassing both the waiting period for consecutively arriving image frames stored in the database server and the inference process.

$$\Upsilon_i^{AIoT} = 1/\left(t_i^d - t_{i-1}^d\right) \quad (1)$$

2) Image frame arrival rate: Measured in frames per second (fps), this metric indicates the rate at which image frames are successfully received by the laptop server and stored in the MySQL. It is calculated as the inverse of the interarrival time of consecutive image frames recorded in the database. This metric reflects the effectiveness of the wireless IoT networks in transmitting image streams, which is critical for ensuring the performance of AIoT systems. After an image is stored in the MySQL, it is immediately processed by the object detection model. The arrival rate (Λ_i) of the image frame *i* can be expressed as Equation (2), where τ_i represents the arrival time of the image frame *i* at the database server, transmitted via wireless IoT networks from the ESP32-CAM to the laptop.

$$\Lambda_i = 1/(\tau_i - \tau_{i-1}) \tag{2}$$

3) Image transmission delay: Measured in seconds, this metric represents the time interval between the initiation of the TCP SYN by the ESP32-CAM, prior to transmitting data segments of an image frame, and the successful reception of the TCP FIN by the laptop after transmitting the final data segments of the image frame. As this delay is measured at the frame level, and each image frame consists of multiple data segments, it captures the impact of missing or retransmitted segments resulting from collisions, contentions, congestions, and wireless errors. It is important to note that this image transmission delay is less than the interarrival time of consecutive image frames received by the destination laptop. This metric directly measures the performance of wireless IoT networks due to the employed MAC and transport protocols. Let D_i, as defined in Equation (3), be the transmission delay of the image frame *i*. It is calculated as the time difference between the initiation time of the TCP SYN $\left(t_i^S\right)$ and the reception time of the TCP FIN $\left(t_i^F\right)$, as monitored using Wireshark at the laptop.

$$D_i = t_i^F - t_i^S \tag{3}$$

4) Image frame size (L_i): This metric, measured in bytes, indicates the size of each image frame generated and successfully transmitted from the ESP32-CAM to the laptop server.

5) Goodput: This metric, measured in bits per second (bps), represents the efficiency of data transmission within the wireless IoT network. It is calculated as the ratio of received data bits, excluding protocol overhead and retransmitted or lost segments, to the interarrival time of consecutive image frames stored in the database server. The goodput (Γ_i) of each image frame *i* is determined as the ratio of the image frame size (L_i) to the interarrival time $(\tau_i - \tau_{i-1})$. This relationship is defined in Equation (4).

$$\Gamma_i = L_i/(\tau_i - \tau_{i-1}) \tag{4}$$

4. Numerical results and discussions

This section presents and analyzes the numerical results of all experiments conducted in this study. Initially, the performance of the object detection rate is examined. Subsequently, the frame size of each image captured by the ESP32-CAM is discussed as a primary input factor for wireless networks. Following this, the performance metrics of the wireless IoT networks are analyzed at the image frame level.

Figure 3 presents the average object detection rates measured under two wireless distance scenarios: 1 meter and 50 meters. These results are influenced by

the laptop specifications detailed in the previous section. In the 50-meter scenario, the average object detection rates range from 0.2 to 1.4 fps, significantly lower than the 2.3 to 8.2 fps observed in the 1-meter scenario. The average object detection rates decrease as the image resolution increases in both wireless distance scenarios. Higher image resolutions demand greater computational and network resources, which impacts the object detection rates. The bar graphs highlight substantial variations in object detection rates across image resolutions, illustrating the dependency of detection rates on both resolution and wireless distance. The subsequent numerical analysis will further explore how wireless IoT networks influence the performance of the AI object detection system.

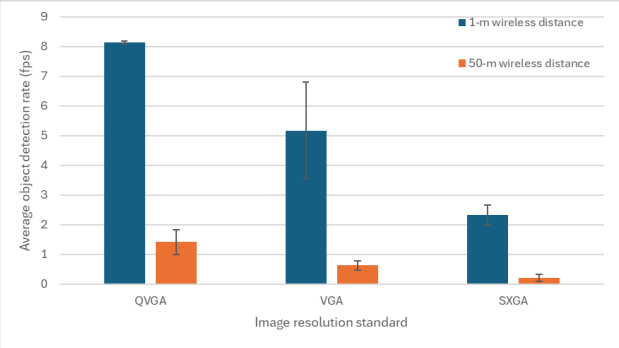


Figure 3 Object detection rate

The QVGA, VGA, and SXGA images successfully transmitted and recorded in the laptop database server during these experiments, as shown in Figure 4, have average frame sizes ranging from 10.7 to 11.1 KB, 27.7 to 29.5 KB, and 101.5 to 105.2 KB, respectively. Similarly, the average size of each image frame is approximately 10 to 100 KB. As illustrated in Figure 4, higher resolution images result in larger data sizes. Consequently, larger image sizes cause greater image transmission delay and increase AI inference time. Therefore, image size is a critical initial factor that influences other response variables in all performance measurements of our AIoT systems.

Figure 4 Average frame size of successful transmitted image to the laptop server

The average network goodput for each image frame transmission is presented in Figure 5. In the 1-meter distance scenario, the average wireless goodput ranges from 0.9 to 2 Mbps, with QVGA having the lowest goodput and SXGA the highest. Higher image resolutions result in higher goodput in the 1-meter wireless distance scenario. In contrast, the average wireless goodput in the 50-meter distance scenario ranges from 0.1 to 0.2 Mbps, with no statistical difference across the image resolution standards. The subsequent numerical analysis will explore the discrepancies observed in the 50-meter wireless distance scenario compared to the 1-meter scenario.

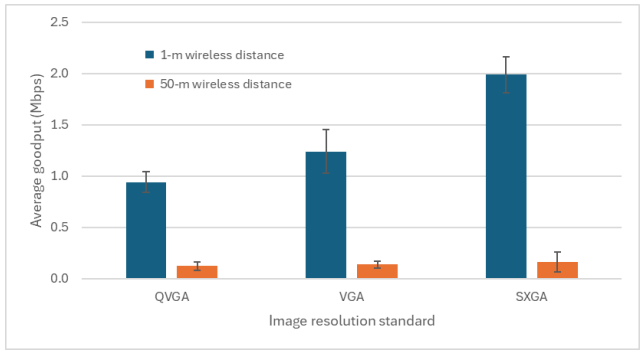


Figure 5 Average end-to end wireless goodput (Mbps) from the ESP-CAM client to the laptop server

The average delay for each image frame transmission between the ESP32-CAM client and the laptop server is depicted in Figure 6. The average delay results (0.1 – 0.3 s) for the 1-meter distance scenario are significantly lower than those for the 50-meter distance scenario (0.5 - 4.4 s). The delay levels increase with the image frame sizes in both wireless distance scenarios, which is expected.

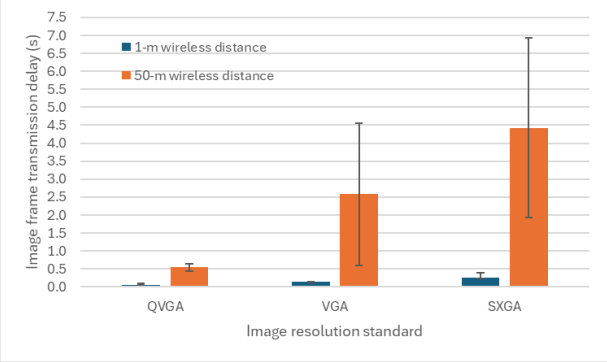


Figure 6 Average end-to-end image frame transmission delay from the ESP32-CAM client to the laptop server

The average delay shown in Figure 6 represents the period encompasses several processes: initiating a TCP connection, successfully transmitting all TCP segments of an image frame, and terminating the TCP connection. The delays exhibit an increasing trend with higher image resolutions. Additionally, the average delays increase substantially when the distance between the camera and the laptop extends from 1 to 50 meters.

The primary contributors to the observed delay results are the image resolution size and the distance between the communication source and destination. Firstly, different image resolution standards result in varying amounts of data bits in each image frame. Higher resolutions produce larger amounts of data. Furthermore, larger image frames must be fragmented into more TCP segments. These segments are subject to congestion, contention, collisions, and retransmissions within wireless multi-hop IoT networks. Consequently, the higher number of segments in larger image frames results in longer transmission times. Secondly, larger radio distances cause greater signal attenuation at the destination, increasing the probability of errors during packet transmission. The increased error rate places additional strain on the wireless network, necessitating retransmissions. Consequently, the combination of packet retransmissions and increased transmission times leads to a significant increase in image transmission delay.

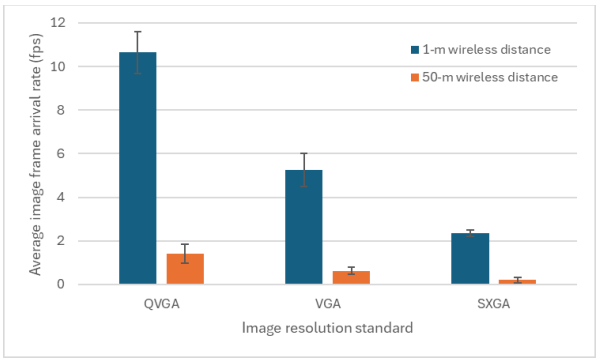


Figure 7 Average image frame arrival rate at the laptop

The next metric to consider is the image frame arrival rate, as depicted in Figure 7. This metric highlights the average incoming frame rate of images for AI object detection. It represents the rate at which image frames are stored in the database server on the laptop and become available for AI object detection. The system reads each frame from the database to process the image using AI object detection algorithms. The numerical results for the average image frame arrival rate in the 1-meter and 50-meter distance scenarios range from 2.4 to 10.6 fps and 0.2 to 1.4 fps, respectively. As shown in Figure 7, these frame arrival

rates decrease as image sizes and wireless distances increase.

As mentioned previously, wireless multi-hop IoT networks can degrade communication performance due to congestion, contention, collisions, and wireless errors. After several retries, it is possible for all TCP segments to be successfully received by the laptop; however, this outcome is not guaranteed. If even one TCP segment is missing, the HTTP POST process for that image transmission will fail, causing the image frame to be discarded at the source. In such cases, the ESP32-CAM captures the next image and initiates a new TCP connection to the laptop to transmit the segments of the new image frame. These phenomena lead to reduced image frame arrival rates in AIoT systems, which in turn can adversely affect the overall efficiency of AI-driven real-time object detection systems.

By treating the incoming image frame arrival rate as the input factor for AI object detection algorithms and the object detection rate as the response variable, the numerical data in Figure 3 and Figure 7 can be analyzed collectively.

Firstly, in the 1-meter wireless distance scenario for QVGA, the average image frame arrival rate is 10.6 fps in Figure 7, which is higher than the average object detection rate of 8.2 fps in Figure 3. These represent the highest frame rates observed across all wireless experiment scenarios, highlighting potential limitations in the object detection system's ability to process all incoming frames at this resolution in real-time processing. Secondly, for VGA and SXGA image resolutions in the 1-meter wireless distance case, the average image frame arrival rates in Figure 7 and the average object detection rates in Figure 3 are equal. Similarly, the results for both metrics are also equal across all image resolutions in the 50-meter wireless distance scenario. This equality suggests that the system's object detection rate is limited by the frame arrival rate in these cases. Overall, these results indicate that frame arrival rates act as a bottleneck for object detection rates in AIoT systems, emphasizing the need for further optimization of network performance and object detection capabilities.

5. Conclusions

This study provides an in-depth analysis of the influence of wireless IoT networks on object detection rates in real-time AIoT systems. The findings highlight that edge devices are not the sole limiting factor for enhancing object detection rates. Wireless IoT networks, particularly in environments prone to congestion, contention, and errors, can significantly constrain the object detection performance of AIoT systems. These insights lay the groundwork for future research aimed at optimizing wireless IoT networks to meet the demanding requirements of real-time AIoT applications.

6. Appendix

In the absence of a wireless IoT network, where image frames are continuously available for inference, the inference rate of object detection $\left(\mathrm{I}_i^{obj\ detect}\right)$ for each image frame *i* is mathematically represented as Equation (5). Let t_i^f and t_i^p denote the time immediately preceding and following the inference process for the image frame *i*, respectively.

$$\mathrm{I}_i^{obj\ detect} = 1/\left(t_i^f - t_i^p\right) \tag{5}$$

Accounting for the impact of IoT networks, the object detection rate Υ_i^{AIoT} is generalized from

Equation (1) into two cases, based on the interplay between Λ_i and $\mathrm{I}_i^{obj\ detect}$.

$$\Upsilon_i^{AIoT} = \begin{cases} \frac{1}{\left(t_i^f - t_i^p\right)} & \Lambda_i \geq \mathrm{I}_i^{obj\ detect} \\ \frac{1}{(\tau_i - \tau_{i-1})} & \text{otherwise} \end{cases} \quad (6)$$

These two cases highlight distinct bottlenecks in AIoT systems. When $\Lambda_i \geq \mathrm{I}_i^{obj\ detect}$, the database queues at least one image frame awaiting the inference process. In other words, the inference time is longer than the interarrival time for each image frame. This implies that the inference time exceeds the interarrival time for each image frame, making AI object detection the primary bottleneck. Conversely, when $\Lambda_i < \mathrm{I}_i^{obj\ detect}$, the database remains empty, and the inference process waits for a new image frame to arrive. This indicates that the inference time is faster than the interarrival time, with the IoT network becoming the limiting factor in the AIoT system's performance.

Next, the impact of network goodput performance on AIoT systems is examined. Specifically, the arrival rate Λ_i can be mathematically expressed in terms of goodput Γ_i, as defined in Equation (7).

$$\Lambda_i = \Gamma_i / L_i \quad (7)$$

The relationships given in Equations (6) and (7) imply that the detection rate Υ_i^{AIoT} can be possibly increased if the goodput Γ_i for a given image size L_i is sufficient to make the frame arrival rate Λ_i equal to the inference rate $\mathrm{I}_i^{obj\ detect}$. However, if the goodput is excessively high and results in the arrival rate exceeding the inference rate, the object detection rate Υ_i^{AIoT} does not increase further. In contrast, if the goodput is insufficient to raise the arrival rate to match the inference rate, the object detection rate Υ_i^{AIoT} will equal the frame arrival rate Λ_i. The goodput is influenced by wireless IoT network performance factors, including contention, collision, and wireless errors. Accordingly, careful design and seamless integration of wireless IoT networks and AI object detection are essential to ensuring the performance and efficiency of cost-effective AIoT systems.

7. References

[1] S. Liu, B. Guo, C. Fang, Z. Wang, S. Luo, Z. Zhou and Z. Yu. "Enabling Resource-Efficient AIoT System With Cross-Level Optimization: A Survey," *IEEE Communications Surveys and Tutorials,* Vol. 26 (1), p. 389–427, 2024. doi: 10.1109/COMST.2023.3319952

[2] F. Oliveira, D. G. Costa, F. Assis and I. Silva. "Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning," *Internet of Things,* Vol. 26, pp. 101153, 2024. doi: 10.1016/j.iot.2024.101153

[3] G. H. Yue, A. Y. Alex Chang and A. Suwanto. "Current Knowledge on the Biology and Aquaculture of the Endangered Asian Arowana," *Reviews in Fisheries Science & Aquaculture,* Vol. 28 (2), pp. 193-210, 2020. doi: 10.1080/23308249.2019.1697641

[4] N. H. Ritonga, A. N. Jati and R. Wijaya, "Automatic arowana raiser controller using mobile application based on Android". *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob). 13-15 September. Bandung, Indonesia* : pp. 63-67, 2016. doi: 10.1109/APWiMob.2016.7811454

[5] Y.-C. Lin, W.-H. Li, C.-C. Lee and L. Y. Y. Lu. "Arowana Breeding System Development Using Internet of Things," *WSEAS Transactions on Computer Research,* Vol. 5, pp. 20-26, 2017.

[6] I. Ranaweera, G. Weerakkody, B. K. Balasooriya, N. S. Swarnakantha and U. Rajapaksha, "Image Processing and IoT-based Fish Diseases Identification and Fish Tank Monitoring System". *International Conference on Advancements in Computing (ICAC). 9-10 December. Colombo, Sri Lanka* : pp. 144-149, 2022. doi: 10.1109/ICAC57685.2022.10025327

[7] L.-B. Chen, Y.-H. Liu, X.-R. Huang, W.-H. Chen and W.-C. Wang. "Design and Implementation of a Smart Seawater Aquarium System Based on Artificial Intelligence of Things Technology," *IEEE Sensors Journal,* Vol. 22 (20), pp. 19908-19918, 2022. doi: 10.1109/JSEN.2022.3200958

[8] K. Sanada, N. Komuro, Z. Li, T. Pei, Y.-J. Choi and H. Sekiya. "Generalized analytical expressions for end-to-end throughput of IEEE 802.11 string-topology multi-hop networks," *Ad Hoc Networks,* Vol. 70, pp. 135-148, 2018. doi: 10.1016/j.adhoc.2017.11.009

[9] B. Scheuermann, C. Lochert and M. Mauve. "Implicit hop-by-hop congestion control in wireless multihop networks," *Ad Hoc Networks,* Vol. 6 (2), pp. 260-286, 2008. doi: 10.1016/j.adhoc.2007.01.001

[10] S. Xu, T. Saadawi and M. Lee, "On TCP over wireless multi-hop networks". *MILCOM. 28-31 October. McLean, VA, USA* : pp. 282-288, 2001. doi: 10.1109/MILCOM.2001.985805

[11] G. Bianchi. "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Area in Communications,* Vol. 18 (3), pp. 535-547, 2000. doi: 10.1109/49.840210

[12] Espressif. (7 November 2024). *Espressif IoT Development Framework*. [Online] Available : https://github.com/espressif/esp-idf

[13] D. Adami, M. O. Ojo and S. Giordano. "Design, Development and Evaluation of an Intelligent Animal Repelling System for Crop Protection Based on Embedded Edge-AI," *IEEE Access,* Vol. 9, pp. 132125-132139, 2021. doi: 10.1109/ACCESS.2021.3114503

[14] P. Adarsh, P. Rathi and M. Kumar, "YOLO v3-Tiny: Object Detection and Recognition using one stage improved model". *International Conference on Advanced Computing and Communication Systems (ICACCS). 6-7 March. Coimbatore, India* : pp. 687-694, 2020. doi: 10.1109/ICACCS48705.2020.9074315

[15] Shenzhen Ai-Thinker Technology. (7 November 2024). *ESP32-CAM Module*. [Online] Available : https://loboris.eu/ESP32/ESP32-CAM Product Specification.pdf