https://doi.org/10.31436/iiumej.v26i3.3433

DESIGNING AND IMPLEMENTING REAL-TIME DEEP LEARNING OBJECT DETECTION IN UNMANNED AERIAL VEHICLES

NURUL FAHMI ARIEF HAKIM [1], ALYANI ISMAIL [2], IWAN KUSTIAWAN [1*], DEASY ROSANTI [3], TAQIYUDDIN YAZID ZAIDAN [1]

[1]*Dept. of Electrical Engineering Education, Universitas Pendidikan Indonesia, Bandung, Indonesia*
[2]*Wireless and Photonic Networks Research Center, Universiti Putra Malaysia, Serdang, Malaysia*
[3]*Electrical Engineering Study Program, Politeknik TEDC, Bandung, Indonesia*

**Corresponding author: iwan_kustiawan@upi.edu*

(Received: 1 September 2024; Accepted: 29 June 2025; Published online: 9 September 2025)

ABSTRACT: The rapid development of surveillance technology is readily apparent, particularly in monitoring vast and remote locations that present difficulties for human accessibility. Within the realm of contemporary surveillance methods, the utilization of Unmanned Aerial Vehicles (UAVs) has garnered a considerable amount of interest. The swift advancements in science and technology have led to the progressive incorporation of Artificial Intelligence (AI) technologies, particularly in tasks such as monitoring and reconnaissance. The results of this research contribute to the creation of a prototype for a UAV capable of conducting autonomous monitoring missions and integrating artificial intelligence technologies for real-time video processing. This study utilized an experimental methodology, and a Raspberry Pi was utilized for artificial intelligence processes integrated with the aircraft controller. During the decisive experiment, the unmanned aerial vehicle UAV could effectively travel to the designated area, reaching an accuracy of 78.6% in its AI processing.

ABSTRAK: Perkembangan pesat teknologi pemantauan kini semakin ketara, terutamanya dalam pemantauan kawasan yang luas dan terpencil sukar diakses manusia. Dalam konteks kaedah pemantauan kontemporari, penggunaan Kenderaan Udara Tanpa Pemandu (UAV) telah menarik minat ramai. Kemajuan pesat sains dan teknologi telah menghasilkan gabungan teknologi Kecerdasan Buatan (AI) secara progresif, terutama dalam bidang pemantauan dan peninjauan. Penyelidikan ini memberi sumbangan kepada penciptaan prototaip UAV yang mampu menjalankan misi pemantauan sendiri, disamping gabungan teknologi kecerdasan buatan bagi memproses video masa nyata. Kajian ini menggunakan metodologi eksperimen dan Raspberry Pi yang disepadukan dengan pengawal pesawat bagi proses kecerdasan buatan. Semasa eksperimen penentuan, UAV berjaya bergerak dengan berkesan ke kawasan yang ditetapkan, mencapai ketepatan 78.6% dalam pemprosesan AI.

KEYWORDS: *Autonomous vehicle, deep learning, object detection, surveillance, UAV.*

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have experienced rapid development and adoption across various sectors, including agriculture, search and rescue, surveillance, transmission line maintenance, infrastructure inspection, environmental mapping, and more [1]. Their flexibility, mobility, and cost-effectiveness have made UAVs indispensable for both civilian and military operations, particularly in ground monitoring and surveillance tasks [2]. Among their key

advantages are real-time video transmission, high-resolution imaging, and the ability to operate in complex or high-risk environments [3]. However, most existing UAV-based surveillance systems are limited by their reliance on manual operation and post-processing, which constrains their ability to deliver timely, autonomous threat detection and response.

As urban environments become increasingly complex and densely populated, there is an urgent need for intelligent, adaptive, and real-time aerial surveillance systems to enhance public safety and situational awareness [4]. Traditional systems often struggle to respond effectively to the dynamic and unpredictable nature of urban threats. Static cameras and manually controlled drones are insufficient for ensuring fast and reliable surveillance, especially in densely populated or hard-to-reach areas [5]. Moreover, with rising incidents of traffic congestion, public safety threats, and emergency events, surveillance platforms must be able to detect and respond to anomalies autonomously. The integration of artificial intelligence into UAVs offers a promising solution, shifting surveillance from passive observation to proactive and autonomous monitoring [6].

Prior studies have explored AI integration in UAVs, such as video surveillance analysis using the Aids to Navigation (ATON) dataset and vehicle speed estimation through deep neural networks [7], [8]. The videos utilized are sourced from the ATON dataset, which comprises numerous surveillance camera and UAV footage that is subsequently processed by AI algorithms. Nevertheless, the level of user practicality is reduced when the AI system is not real-time and is applied while the UAV is in operation. This research addresses that need by developing an AI-enhanced UAV system capable of both manual and autonomous flight, designed explicitly for real-time surveillance and object detection. By equipping the UAV with a microcomputer and onboard camera, the system integrates AI algorithms to automatically detect and recognize objects in real time, particularly near residential zones and roadways. Unlike conventional systems that require manual monitoring or offline video analysis [9], the proposed system functions autonomously and processes visual data onboard, reducing latency and operator dependency.

The uniqueness of this article is its comprehensive integration of autonomous navigation, integrated AI-powered visual processing, and real-time object recognition within a single UAV platform. This method not only enhances the current capabilities of UAVs but also addresses the urgent need for intelligent, self-sufficient aerial monitoring solutions. The contribution of this article is to introduce a practical, scalable, and cost-effective UAV surveillance framework that utilizes lightweight AI models and accessible hardware components, making it feasible for broader adoption in various applications. The software is also open-source, allowing everyone to contribute to its development. Moreover, it enhances the state-of-the-art UAV-based surveillance by enabling proactive monitoring and immediate object detection without ground-based computation or human intervention. By bridging the gap between UAV mobility and artificial intelligence, this study contributes to advancing intelligent sensing technologies, with significant implications for innovative city development, disaster response systems, and real-time traffic and security monitoring.

2. METHOD

The methodology for designing and implementing UAVs follows the waterfall approach, as illustrated in Fig. 1. This approach adheres to a pre-established sequence and linear progression, where each phase builds upon the previous one. The UAV's design process begins with gathering requirements to ensure the UAV conforms to specified criteria. In the second phase, the system design occurs, encompassing the architecture of the UAVs, including

https://doi.org/10.31436/iiumej.v26i3.3433

hardware, software, and integration with a deep learning object detection system. During the implementation phase, the hardware and software components of the UAV are constructed and developed. Using 3D design software, the UAV's frame is built, taking into account the flight controller, GPS, sensors, cameras, motors, and propellers. Concurrently, the deep learning object detection system is integrated with the onboard hardware for real-time processing. Subsequent phases focus on constructing a three-dimensional model of the UAVs and designing hardware and software systems, ensuring the integration of all essential components. The deep learning procedure captures object images to generate training data for machine learning models. A functional prototype is then constructed by integrating UAV subsystems and object detection systems, enabling the UAVs to navigate autonomously during testing. Once the system integration is complete, the UAV prototype will detect objects during monitoring missions near residential areas or roadways. The system's performance is thoroughly evaluated, and predetermined criteria are applied to ensure its capability for monitoring aerial objects and the proper functioning of the deep learning program.

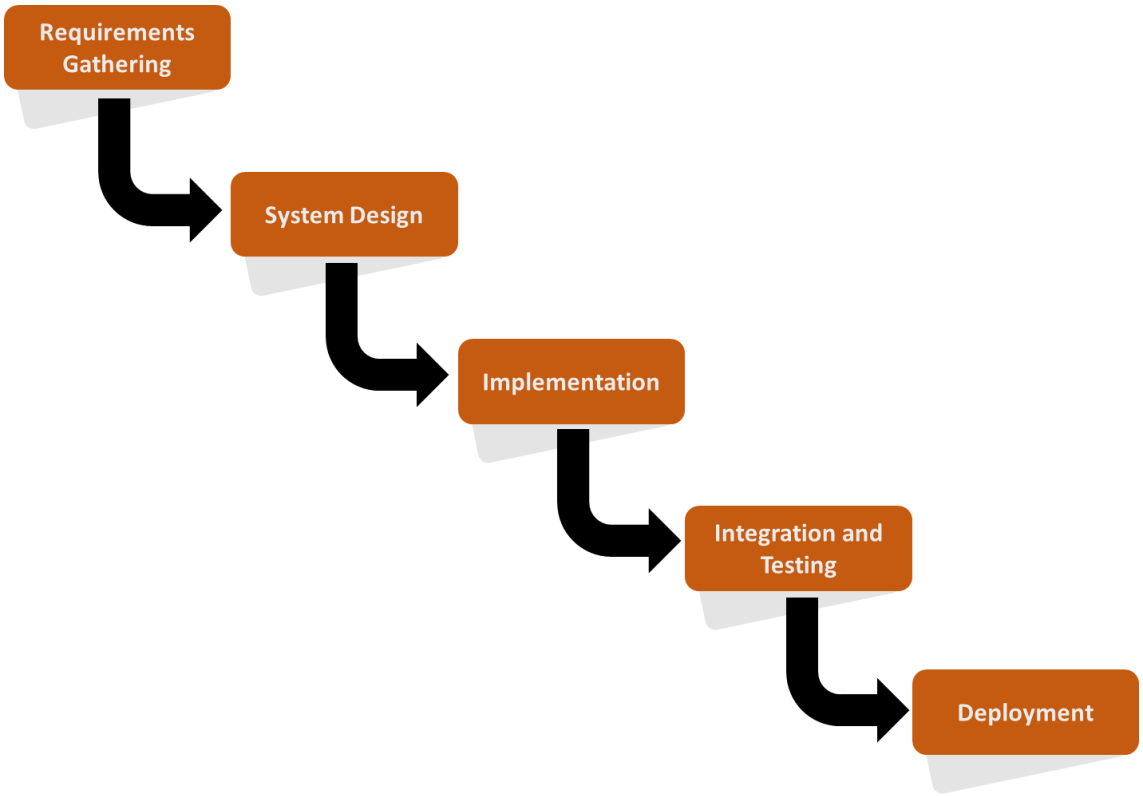


Figure 1. UAVs design flowchart

2.1. UAVs design

The proposed UAV design features the F330 frame, which has a diagonal length of 330 mm and a height of 70 mm. To accommodate the addition of an AI system, extra design parts will be implemented on the legs, resulting in an increased height of the UAVs [10]. The body of the UAV design is illustrated in Fig. 2. In the hardware design, this UAV's control system utilizes the Pixhawk Flight Controller. The Pixhawk controller features a built-in inertial measurement unit (IMU) module with four sensors: a magnetometer, a gyroscope, a barometer, and an accelerometer. In addition to the Pixhawk controller, the UAV's frame is equipped with GPS, a radio control receiver, a battery, a camera, and other components. The types of UAV system components and their respective positions are detailed in Table 1.

https://doi.org/10.31436/iiumej.v26i3.3433

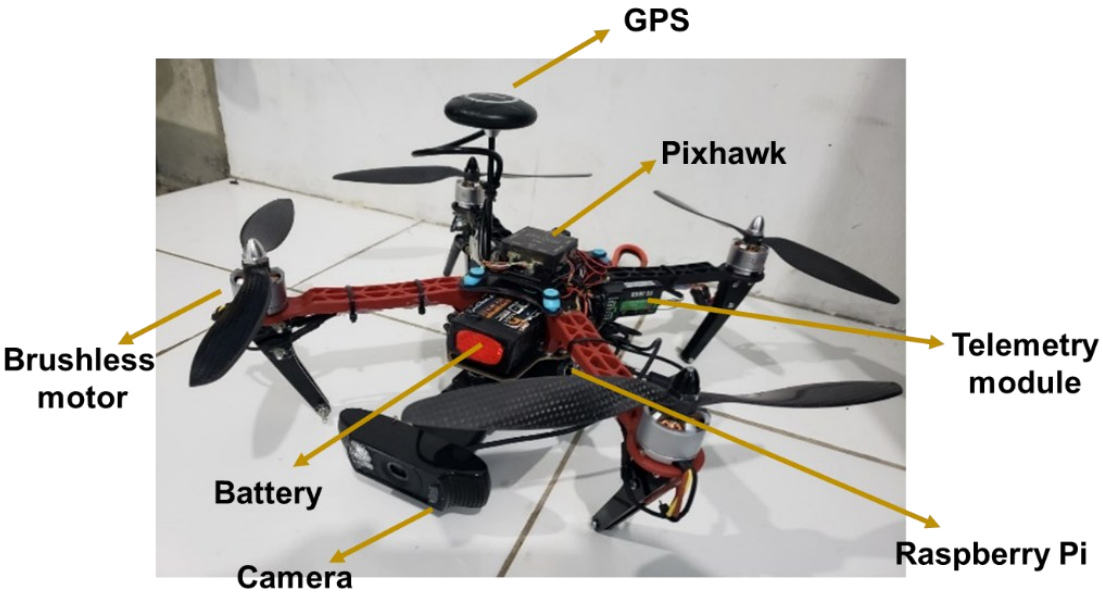


Figure 2. The design of the part increases the height of the landing

Table 1. UAV quadcopter components

Components	Description	
	Type	Position
Frame	F330 quadcopter frame	Main
Flight Controller	Mini Pixhawk PX4	Top layer (center)
GPS	Neo GPS M8N Module	Top layer (center with holder)
Radio Control Receiver	FLSky I6S	Bottom layer (rear left)
Battery	Dinodigy 5000mAh 3S 11,1 Volt 65C	Middle (between the top and bottom layers)
Electronic Speed Controller	4 ESCs BLHeli 25A	Each arm (bottom)
Telemetry for flight data acquisition	2.433 MHZ Radio Telemetry 500mA	Bottom layer (rear right)
Motor	4 Brushless motor 2212 920KV	Each arm
Propeller	4 Propeller Carbon 9047	Each arm
Single Board Computer	Raspberry Pi 4 Model B	Bottom layer (center)
Camera	Logitech C920 Webcam	Bottom layer (front)
Miscellaneous	RC PPM encoder	Ground Control Station
	Power module	Bottom layer
	Buzzer	Top layer
	Pixhawk damping plate	Top layer
	GPS folding antenna	Top layer
	Camera shock absorber	Bottom layer

The UAV hardware block diagram is shown in Fig. 3. The UAV's hardware design centers around two control systems: the Pixhawk, which handles flight processing, and the Raspberry Pi, which manages AI processing. The Pixhawk control system is connected to the GPS, which provides navigation for the UAV. Integrated GPS/INS (Inertial Navigation System) navigation is the standard approach in integrated navigation research. Additionally, the Pixhawk is connected to four ESCs (Electronic Speed Controllers) that regulate four brushless DC motors. Unlike traditional DC motors, brushless DC motors do not have brushes, offering benefits such as high efficiency, speed, torque, and easy maintenance [11]. The Pixhawk also communicates

https://doi.org/10.31436/iiumej.v26i3.3433

with the Ground Control Station (GCS) via telemetry. The GCS is the central hub for UAV ground control and is the only means ground personnel can communicate with the UAV. It functions as a control system, monitoring the UAV and creating flight missions to enable autonomous operation. Furthermore, the Pixhawk is connected to a remote receiver module, which receives input signals from the radio control during manual flights.

Mission Planner is a ground control software frequently employed to establish communication and configuration with Pixhawk, an open-source flight controller for unmanned aerial vehicles (UAVs). Mission Planner allows the UAV to operate autonomously when embedded and configured correctly, executing predetermined flight missions without real-time manual control. Mission Planner is embedded with Pixhawk and is utilized as an autonomous UAV system. Based on real-time sensor inputs (GPS, accelerometer, gyroscope), the flight controller controls the quadcopter in autonomous mode to follow a pre-programmed flight path or execute tasks such as waypoint navigation, takeoff, landing, or hovering. This configuration enables the UAV to function autonomously, even in environments where human control may be challenging or inefficient. An AI algorithm is created to process the video or sensor data of the UAV using the Python programming language once reliable data has been obtained. The AI model can be customized to perform various duties, including object detection, tracking, and classification, contingent upon the mission's objectives. This flight data undergoes processing for filtering and synchronization [12]. The development of the AI model is conducted using Google Colaboratory (Colab), a free cloud-based environment that enables researchers to write and execute Python code in the browser. This is to train and validate the model [13].

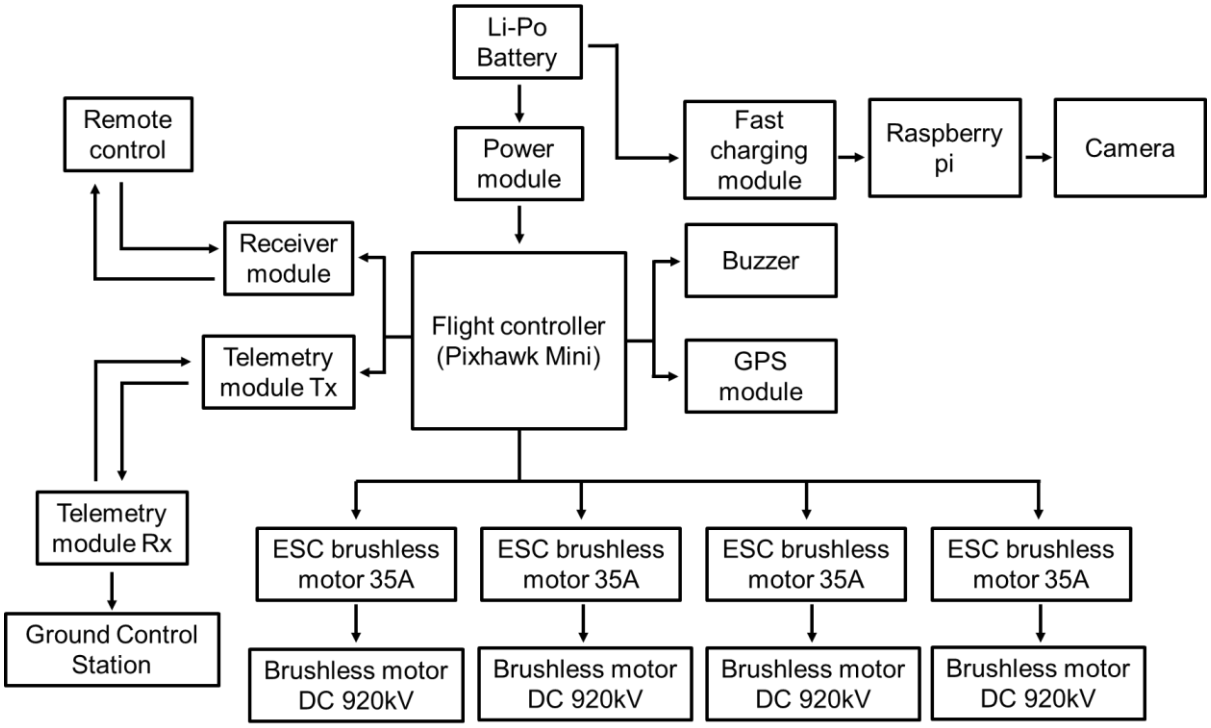


Figure 3. Block diagram of the hardware UAV

2.2. Object detection system

For arithmetic calculations in machine learning, Google has developed an open-source tool called TensorFlow. TensorFlow offers significant advantages in terms of high accessibility and flexibility, with ample resources available on GitHub and online forums to enhance its

https://doi.org/10.31436/iiumej.v26i3.3433

efficiency. Currently, Google provides several pre-trained models on the official TensorFlow website, and MobileNets is one such model trained within TensorFlow. MobileNets is a convolutional neural network (CNN) architecture designed to address the need for efficient computing resources [14]. TensorFlow facilitates transfer learning, a novel machine learning approach that leverages existing knowledge, learning from one environment to solve new, different, but related problems. This is particularly valuable when there is insufficient training data or a need to train a custom model without starting from scratch [15]. The object detection system developed in this research utilizes the TensorFlow API. In this study, researchers employed transfer learning methods to train a custom dataset using the MobilenetV2 model, creating a customized machine-learning model. The objects targeted for training comprise four classes: cars, motorcycles, trees, and trucks.

2.3. Data Preparation

The utilized data comprises a collection of images captured from both cell phone and UAV cameras. Examples of image data are presented in Fig. 4. These images are extracted into frames in *JPG format. They will serve as samples in image processing for detecting roads, cars, motorcycles, trees, and trucks. The total number of collected images amounts to 1,400. Each image will undergo labeling for object detection, as illustrated in Fig. 5. Data labeling has been a focal point in machine learning research. Subsequently, after labeling, the data will be partitioned into 80% training data, 10% validation data, and 10% test data.

Figure 4. Sample picture on the main road

https://doi.org/10.31436/iiumej.v26i3.3433

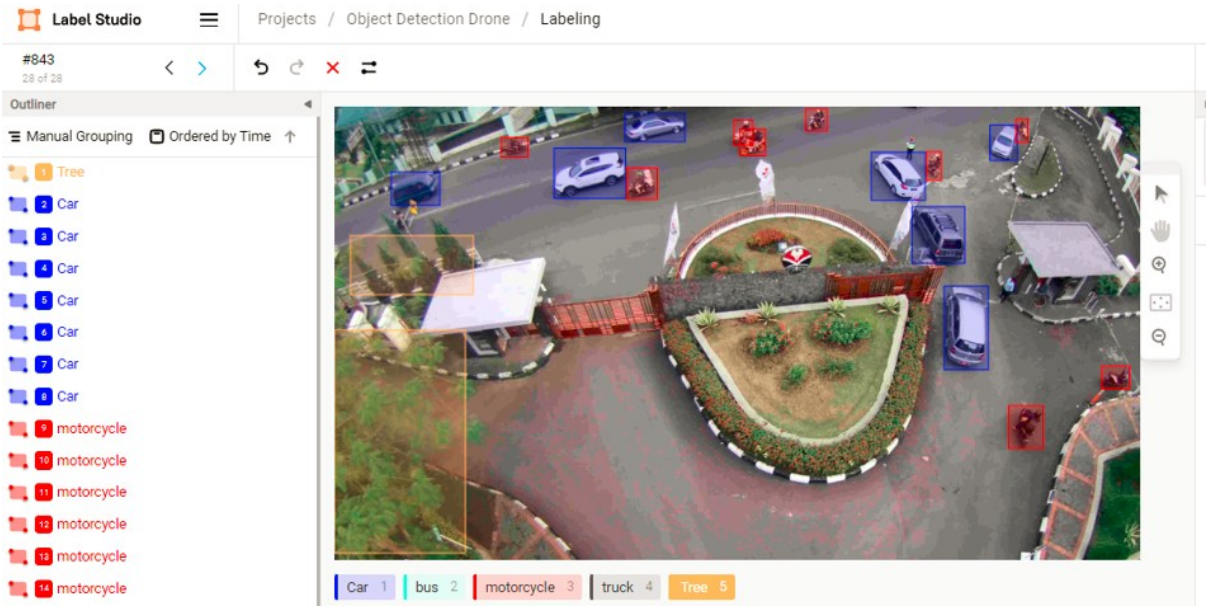


Figure 5. Data labeling using Label Studio

2.4. Deep learning

Deep learning is a machine learning algorithm component used to manage large datasets. Deep learning is swiftly expanding and gaining momentum to resolve digital image and video issues. In recent years, deep learning has been implemented to facilitate the classification of images and the detection of objects. Equations (1) and (2). The object detection system on a UAV will be developed using a deep learning algorithm. The TensorFlow API will be employed by the object detection system developed in this study to generate a deep learning model. The transfer learning method trains a custom dataset from the MobilenetV2 model into a new custom deep learning model.

The construction of a deep learning model is divided into three phases, as illustrated in Figure 6. The following stages are as follows: data preparation, training, and testing [16]. The data preparation stage is the initial phase of the deep learning training process. These stages encompass data acquisition, labeling, LabelMap configuration, and converting the data label results into a pipeline for processing in the subsequent stage. Additionally, the training process stage will require significant time due to the computer's learning process. This stage comprises the convolution, pooling, and activation ReLU stages, which connect all process layers and generate a classification model. The loss value will be determined after the machine's learning or training process. If the Loss value is suitable, the process will proceed to the subsequent stage. If the results are unsatisfactory, the training process stage will be reverted to the training stage by adjusting several parameters to achieve the desired outcome. During the assessment process, the model undergoes a trial by inputting the resulting images and videos. Object detection results will be visible after the testing procedure, allowing conclusions to be drawn.

$$Accuracy = \frac{Images\ Validation\ (True\ Detected\ Object)}{Number\ of\ Total\ Images\ Validation}) \quad (1)$$

$$Loss = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{k}(Images\ Train - Images\ Validation)^2 \quad (2)$$

https://doi.org/10.31436/iiumej.v26i3.3433

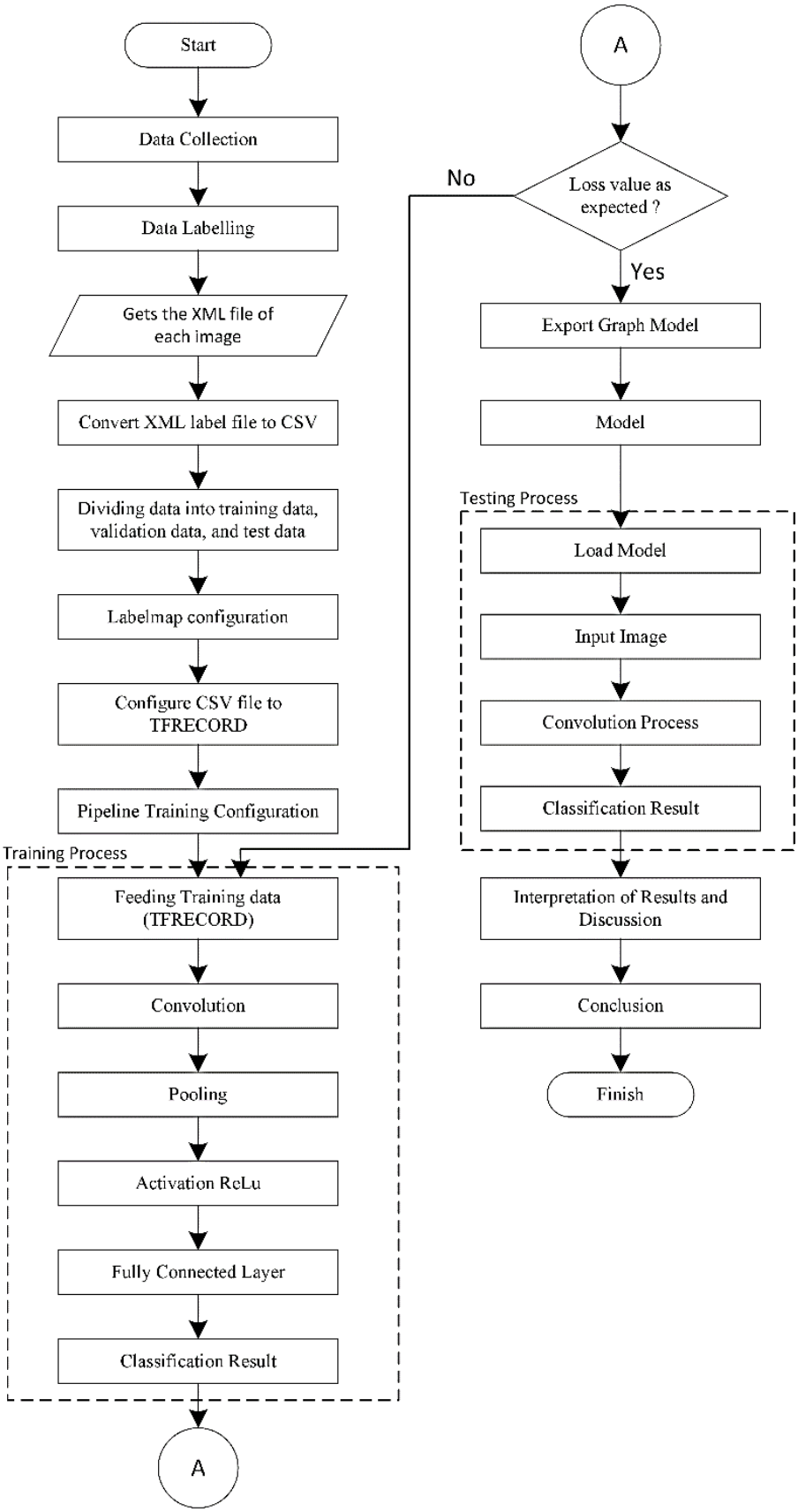


Figure 6. Deep learning flowchart

https://doi.org/10.31436/iiumej.v26i3.3433

3. RESULT AND DISCUSSION

The data taken in this study are in the form of traffic videos on Setiabudhi Street, Bandung, traffic in the Cimahi residential area, and flyovers. After the data is collected, labeling and training are carried out to get the desired machine-learning model. The result of the labeling is in the form of an XML file, which contains the coordinates and bounding box characteristics of each object from each image, as shown in Fig. 7. The number of images taken is 1400. Each image will undergo labeling and subsequent division into 80% (1120 images) for training data, 10% (140 images) for validation data, and 10% (140 images) for test data. The actual images used to train the model are called training data. The neural network is fed images from the "train" set at each training phase. The network forecasts the class and location of objects in the image. The training algorithm modifies the network weights through back-propagation and calculates the loss, which indicates the degree of "loss" in the predictions. The training algorithm utilizes validation data, which comprises images from a "validation" set, to monitor training progress and adjust hyperparameters, including learning rate. Validation images are employed intermittently during training, typically once every specified number of training steps, in contrast to the images in the "train" set. The test data comprises images that the neural network has never encountered during training. These images are intended for human inspection to conduct the final testing of models and assess their accuracy.

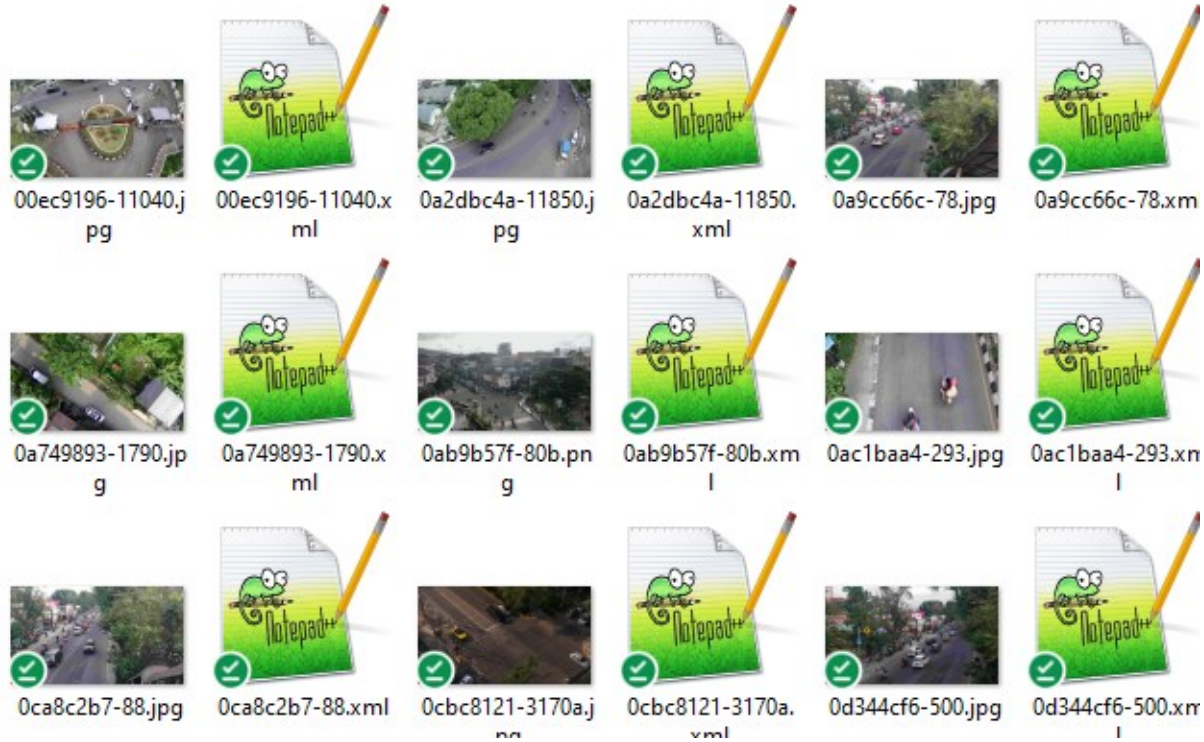


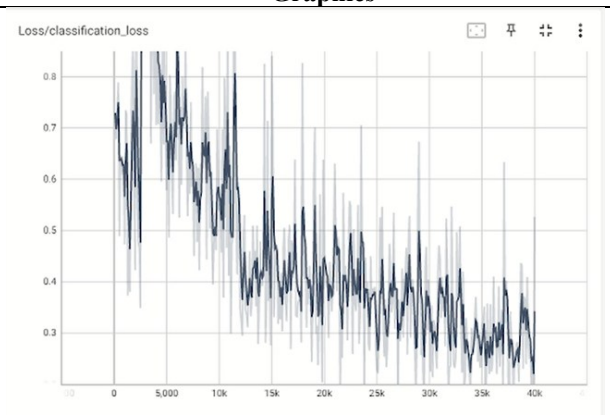
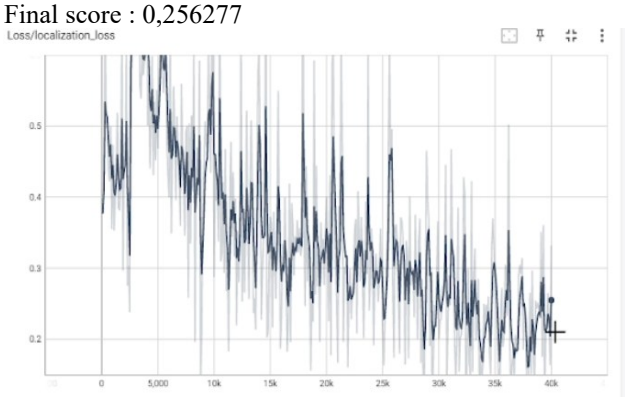
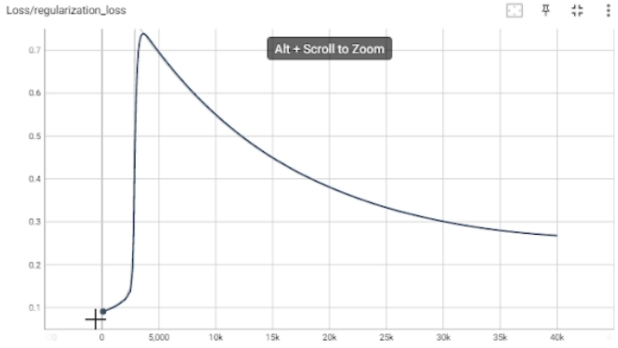
Figure 7. Labelling Result JPG and XML files for each image

In this AI algorithm, objects have been identified, and a label map needs to be created. The label map lists predefined classes: car, motorcycle, tree, and truck. Additionally, each data point in the training and validation sets will be compiled into a .csv file containing all images and their annotations, creating a TFRecord. The TFRecord file is designated for the training and validation datasets, alongside the label map.pbtxt file, encompassing label maps in various formats. Once the data is prepared for training, the training configuration for the SSD-MobileNetV2 model is established, which will be used for training.

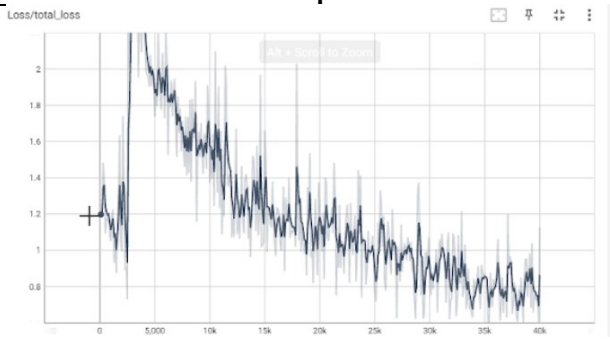
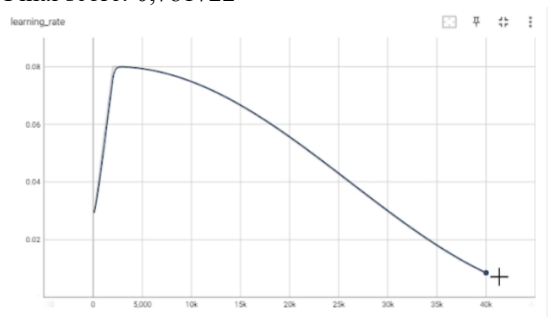
https://doi.org/10.31436/iiumej.v26i3.3433

When the configuration file is prepared, the subsequent stage is to specify the 'batch_size' and 'num_step'. The model will be trained using a total of 'num_step' steps. It is advised to commence with 40,000 steps. The duration of the training increases as the number of strides increases. The quantity of images employed in each training phase is denoted by 'batch_size'. The model can be trained in fewer stages with a larger batch size; however, the size is constrained by the available GPU memory for training. Additionally, the object detection training is prepared for execution. The training process will take 2-6 hours, depending on the GPU used for training, model, 'batch_size', and 'num_step', and it utilizes scripts from the TF Object Detection API. Table 2 displays the training results after the model has been effectively trained. The training results on the graph indicate that the loss is reduced and the information is more detailed as the number of steps increases.

Table 2. Training Result

Loss	Graphics	Description
Classification Loss	 Final score : 0,256277	Classification Loss shows whether the bounding box class matches the predicted class.
Localization Loss	 Final score : 0,257642	Localization Loss calculates the difference between bounding box predictions and labels.
Regularization Loss	 Final score: 0,267803	The network regularization function generates Regularization Loss and helps point optimization algorithms in the right direction.

https://doi.org/10.31436/iiumej.v26i3.3433

Loss	Graphics	Description
Total Loss	 Final score: 0,781722	Total Loss is the sum of the three previous Losses
Learning Rate	 Final score: 0,008266	The Learning Rate controls how much the weights are updated according to the estimated error

The results of object detection are presented in Figs. 8 and 9. Objects in images and videos are successfully identified with bounding boxes, image labels, and the percentage of image similarity. Based on the detection results, the AI algorithm does not detect several objects. Factors that affect the error include the object being tiny and other objects blocking the view of the object. Furthermore, the training model will be tested in real time when the UAV operates on a monitoring mission. carrying out the mission, it is necessary to make waypoints as shown in Fig.10. The waypoint navigation system is crafted to enable the drone to identify its position and gauge the distance it has covered, enhancing precision in reaching the designated destination along the predetermined route set by the operator.

Figure 8. Object detection from a picture: UAV detected 7 cars, and 1 car was not detected

https://doi.org/10.31436/iiumej.v26i3.3433

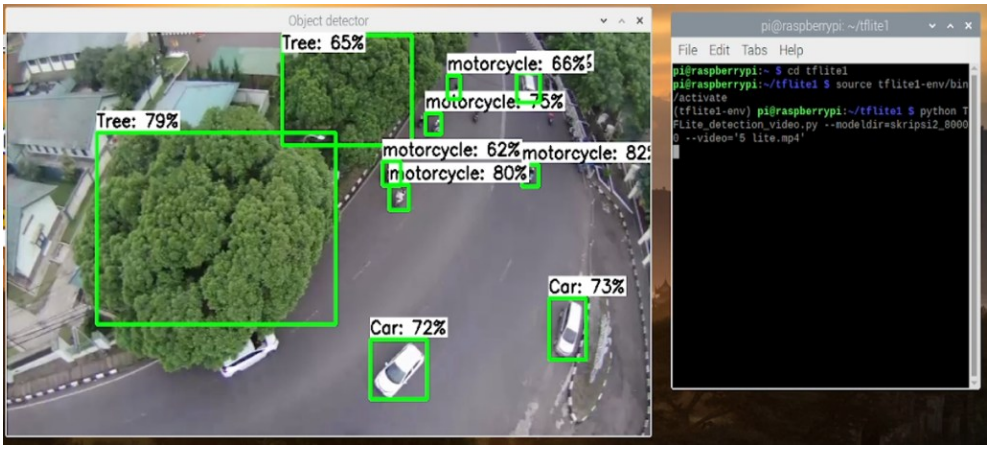


Figure 9. Object detection from video: UAV detected 2 cars, 6 motorcycles, 2 trees, and 1 car was not detected because it was covered by trees

Figure 10. Autonomous flying missions

The results of the autonomous UAV's real-time surveillance mission, specifically related to object detection, are illustrated in Figs. 11 and 12. The UAV descended safely to the last waypoint, exhibiting real-time object detection results aligned with those detected in image and video analysis. Successful object identification during the autonomous UAV mission is indicated by bounding rectangles containing labels specifying the class and the similarity percentage. It is worth mentioning that the Raspberry Pi, which is utilized to process AI for real-time object detection, exhibits a frames-per-second (FPS) value falling below 2. Standard video, on the other hand, generally attains 20-30 FPS. The inconsistency emerges because the Raspberry Pi, which powers the AI algorithm that processes the video, lacks a Graphics Processing Unit (GPU), an essential element in image processing.

Accuracy is assessed throughout the testing process by comparing real objects and those detected by the program. Achieving an accuracy of 78.6%, the object detection algorithm accurately classified 96 out of a possible 122 objects in an image comprising automobiles, motorcycles, trees, and vehicles during a test. The total number of objects was 122. Object detection errors can be caused by a multitude of factors, including the color of the vehicles.

https://doi.org/10.31436/iiumej.v26i3.3433

The absence of color diversity among the vehicles utilized as objects in this study has led to the need for more varied datasets. Additionally, vehicles positioned so far away that they appear as specks were labeled during image labeling, resulting in the model being trained to identify objects whose labels did not correspond to the actual ones.

Figure 11. The UAV detected 2 cars and 1 tree

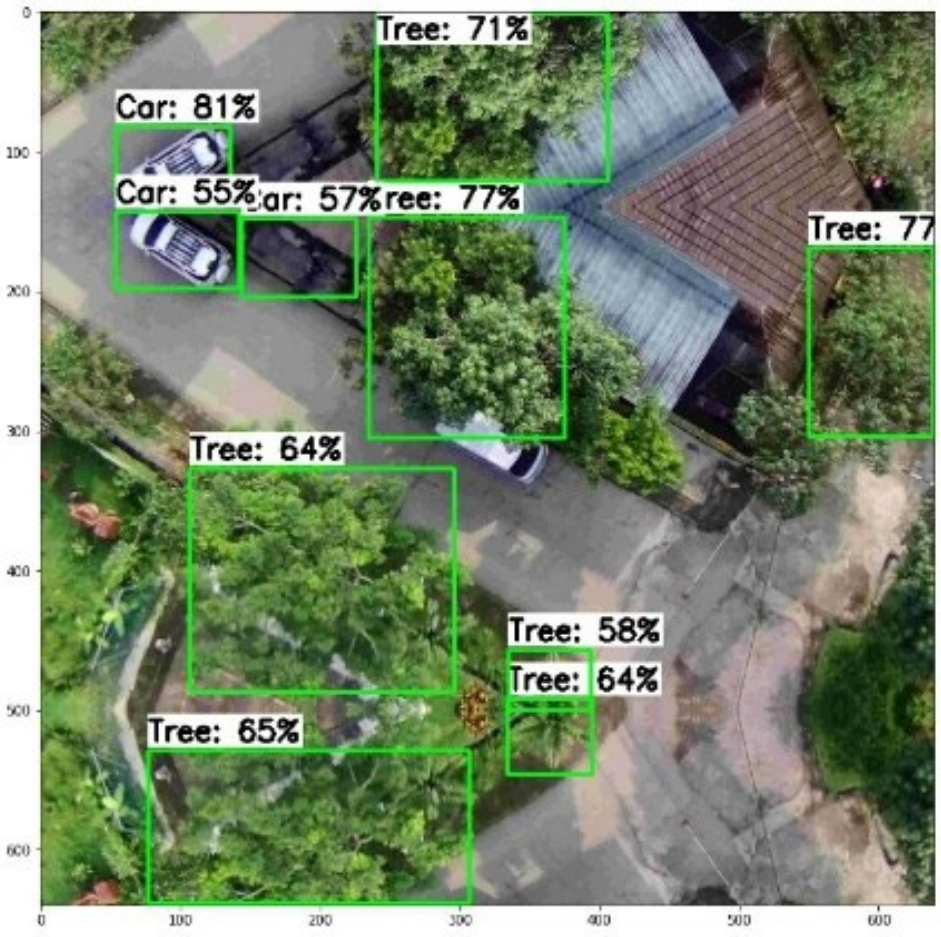


Figure 12. The UAV detected 3 cars and 7 trees

https://doi.org/10.31436/iiumej.v26i3.3433

Integrating Amazon Web Services (AWS) for real-time UAV monitoring demonstrates significant potential for advancing image sensing technology in today's trends and challenges. In the current landscape, UAVs with advanced imaging sensors are widely used for applications such as precision agriculture, environmental monitoring, disaster response, and infrastructure inspection. Fig.13 illustrates the website's monitoring capabilities. The website displays coordinate data that indicates the UAV's altitude is 30.64 meters, which is consistent with the mission. The longitude and latitude of the UAV are also indicated. The UAV has a ground speed of 2.247 and an air speed of 3.579. The currently displayed mode is also suitable, specifically AUTO Mode.

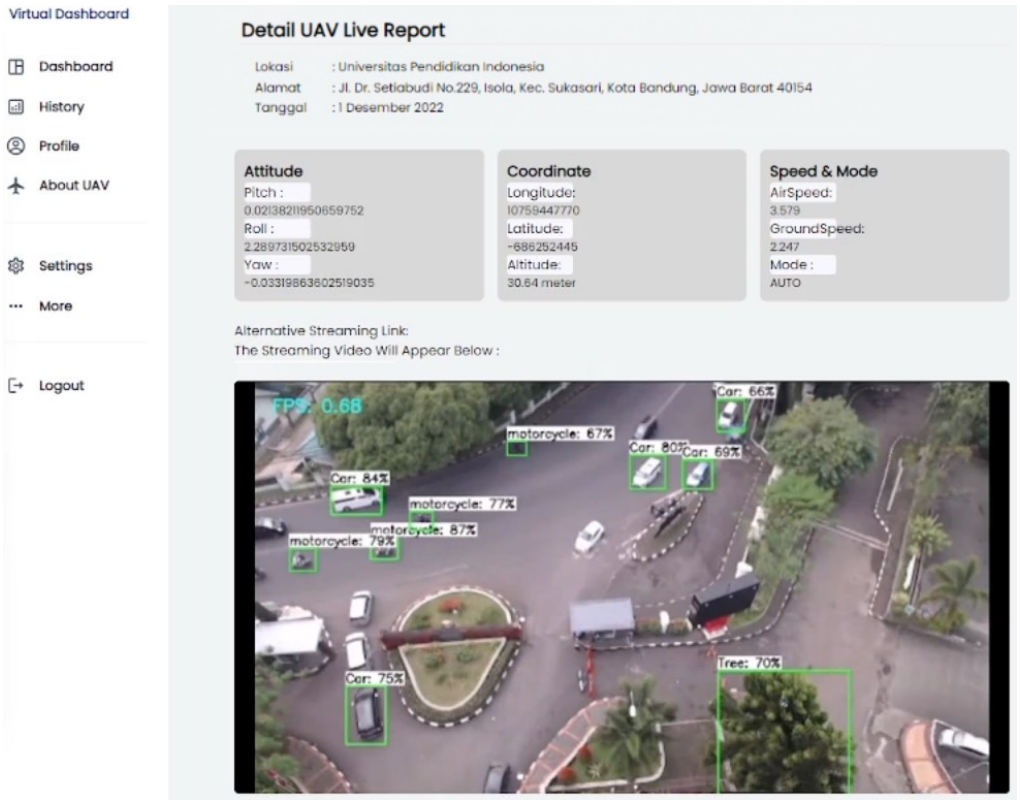


Figure 13. UAVs' real-time monitoring website

4. CONCLUSION

Real-time object detection utilizing deep learning and custom datasets has been successfully developed. The UAV successfully navigated to the last waypoint while simultaneously conducting AI processes. The trained model has proven effective in predicting objects in images and videos captured by the UAV and can operate in real-time during UAV operations. The actual detection rate reaches only 78.6% when the UAV is used for object detection. Detection errors stem from limited datasets and imperfect labels. A more substantial GPU is required to enhance AI processing performance in real-time (increase FPS). For instance, one could integrate a USB Coral for an external GPU with a Raspberry Pi, or opt for a more powerful single-board computer, such as the Nvidia Jetson.

REFERENCES

[1] S. Byun, I.-K. Shin, J. Moon, J. Kang, and S.-I. Choi, “Road Traffic Monitoring from UAV Images Using Deep Learning Networks,” Remote Sens (Basel), vol. 13, no. 20, p. 4027, Oct. 2021, doi: 10.3390/rs13204027.

[2] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends," Intell Serv Robot, vol. 16, no. 1, pp. 109–137, Mar. 2023, doi: 10.1007/s11370-022-00452-4.

[3] A. Fascista, "Toward Integrated Large-Scale Environmental Monitoring Using WSN/UAV/Crowdsensing: A Review of Applications, Signal Processing, and Future Perspectives," Sensors, vol. 22, no. 5, p. 1824, Feb. 2022, doi: 10.3390/s22051824.

[4] J. C. N. Bittencourt, D. G. Costa, P. Portugal, and F. Vasques, "A Survey on Adaptive Smart Urban Systems," IEEE Access, vol. 12, pp. 102826–102850, 2024, doi: 10.1109/ACCESS.2024.3433381.

[5] I. Mademlis, C. Symeonidis, A. Tefas, and I. Pitas, "Vision-based drone control for autonomous UAV cinematography," Multimed Tools Appl, vol. 83, no. 8, pp. 25055–25083, Aug. 2023, doi: 10.1007/s11042-023-15336-7.

[6] N. Jain, A. Gambhir, and M. Pandey, "Unmanned Aerial Networks—UAVs and AI," 2024, pp. 321–351. doi: 10.1007/978-981-97-6790-8_12.

[7] M. T. Nguyen, L. H. Truong, and T. T. H. Le, "Video Surveillance Processing Algorithms utilizing Artificial Intelligent (AI) for Unmanned Autonomous Vehicles (UAVs)," MethodsX, vol. 8, p. 101472, 2021, doi: 10.1016/j.mex.2021.101472.

[8] A. Gomaa, T. Minematsu, M. M. Abdelwahab, M. Abo-Zahhad, and R. Taniguchi, "Faster CNN-based vehicle detection and counting strategy for fixed camera scenes," Multimed Tools Appl, vol. 81, no. 18, pp. 25443–25471, Jul. 2022, doi: 10.1007/s11042-022-12370-9.

[9] C. Chalmers, P. Fergus, C. A. Curbelo Montanez, S. N. Longmore, and S. A. Wich, "Video analysis for the detection of animals using convolutional neural networks and consumer-grade drones," J Unmanned Veh Syst, vol. 9, no. 2, pp. 112–127, Jun. 2021, doi: 10.1139/juvs-2020-0018.

[10] T. Y. Zaidan, N. F. A. Hakim, I. Kustiawan, A. M. Ridwan, S. A. T. Al Azhima, and I. Fahmi, "Video Live Streaming System on Unmanned Aerial Vehicles (UAVs) Using Autonomous Waypoint Navigation to Support Traffic Monitoring System," in 2023 9th International Conference on Wireless and Telematics (ICWT), IEEE, Jul. 2023, pp. 1–5. doi: 10.1109/ICWT58823.2023.10335464.

[11] S. Kumar Vishwakarma and M. Rahman, "Speed Control of PV Array-Based Z-Source Inverter Fed Brushless DC Motor Using Dynamic Duty Cycle Control," in 2022 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), IEEE, Jun. 2022, pp. 1–6. doi: 10.1109/ICEFEET51821.2022.9847935.

[12] D. F. Carlson and S. Rysgaard, "Adapting open-source drone autopilots for real-time iceberg observations," MethodsX, vol. 5, pp. 1059–1072, 2018, doi: 10.1016/j.mex.2018.09.003.

[13] Dr. S. R. Sukhdeve and S. S. Sukhdeve, "Google Colaboratory," in Google Cloud Platform for Data Science, Berkeley, CA: Apress, 2023, pp. 11–34. doi: 10.1007/978-1-4842-9688-2_2.

[14] M. Dou, Z. Hong, and M. Shi, "An Improved Efficient Convolutional Neural Network for Weed Seedlings Detection," in 2021 International Conference on Culture-oriented Science & Technology (ICCST), IEEE, Nov. 2021, pp. 285–289. doi: 10.1109/ICCST53801.2021.00067.

[15] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," IEEE Trans Knowl Data Eng, vol. 33, no. 4, pp. 1328–1347, Apr. 2021, doi: 10.1109/TKDE.2019.2946162.

[16] S. K. Baduge et al., "Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications," Autom Constr, vol. 141, p. 104440, Sep. 2022, doi: 10.1016/j.autcon.2022.104440.