Evaluation of Real-Time Train Overhead Line Component Detection on Edge Device

Yuto Yokomine
Graduate School of Science and Engineering,
Department of Electrical, Electronics, and
Communication Engineering,
Tokyo City University
Tokyo, Japan
g2381278@tcu.ac.jp

Nico Surantha
Faculty of Science and Engineering, Department of
Electrical, Electronics, and Communication
Engineering
Tokyo City University
Tokyo, Japan
nico@tcu.ac.jp

***Abstract*— To ensure the safety of train operations, regular inspections and early detection of damage are essential. Currently, inspections are conducted visually by humans during nighttime hours, which incurs significant costs and time. As a solution, the establishment of automated inspection technology using drones is being explored. In this study, as part of this initiative, the focus was on detecting railway overhead equipment using edge devices and improving detection speed. Building upon prior research involving YOLO and ONNX, a dataset was created for training the AI, which was then used to detect railway overhead equipment on a Raspberry Pi. The trained models achieved mAP and F1-scores of 0.959 and 0.96 for Yolov7, and 0.956 and 0.94 for Yolov7-tiny, respectively. Detection times on the Raspberry Pi were 16.43 seconds for Yolov7 and 2.74 seconds for Yolov7-tiny.**

Keywords—drone, object detection, Yolov7, Yolov7-tiny

I. INTRODUCTION

Currently, in Japan, inspections of railway overhead line equipment are conducted at night to avoid affecting train operations. Railway overhead line equipment primarily consists of three main components: hangers, insulators, and connectors, all of which require regular inspection. In some sections, the equipment has been in use for over 50 years since its installation, leading to significant deterioration. Therefore, periodic inspections are essential for ensuring safe train operations. These inspections are typically carried out at night through visual checks by human inspectors. However, due to the limited time available during nighttime operations, there is a risk of errors. While some sections are inspected using cameras mounted on vehicles, fault detection in these cases still relies on human visual analysis of the recorded footage. Fully automated fault detection systems have yet to be developed. Inspection of railway overhead line equipment is crucial for ensuring safe train operations. However, given Japan's declining birthrate, aging population, and shrinking workforce, securing the necessary personnel for these inspections is expected to become increasingly challenging in the future [1].

To address these challenges, this study focuses on developing a system for automatic fault detection of railway overhead line equipment using drones [2]. Specifically, the study emphasizes object detection of overhead line equipment as part of this broader initiative. If automated drone-based inspections are realized, it is anticipated that the cost of human resources for daily inspections can be reduced, and errors caused by human visual checks can also be mitigated.

Drone inspections can be conducted by capturing footage with drones and applying object detection AI to identify equipment. Furthermore, equipping drones with onboard object detection AI enables them to perform the entire process—from capturing footage to detection—independently. This minimizes the need for communication infrastructure, allowing inspections even during disasters or situations with limited communication capabilities. For real-time drone-based inspections, fast object detection is essential. Thus, accelerating inference speed is critical for enabling immediate object recognition from video footage captured by drones.

One of the current social problems is that many cities in developing countries, despite rapid population growth, are unable to build railroads and rely on automobiles for their transportation infrastructure, resulting in serious traffic congestion. In many cases, the development of transportation infrastructure has been delayed because of the inability to build railroads due to cost issues and a lack of skilled human resources [3]. Using inexpensive drones to inspect train line equipment could solve the cost and human resource problems associated with inspections, recover the cost of operating a railroad, and help reduce inequality in transportation infrastructure. The advantages of these edge device inspections are that they are inexpensive and reduce the number of personnel required.

The detection of railway overhead equipment was realized using the object detection method You Only Look Once (YOLO), which, as demonstrated in prior research, offers higher accuracy and faster inference compared to other methods such as R-CNN and SSD. YOLO is particularly suited for real-time detection [4][5][6]. For real-time inspections using drones, it is necessary to deploy object detection AI on drones, which requires integrating the AI into microcomputers. This study aims to measure the time required for equipment detection on microcomputers and improve its speed.

In conclusion, this study aims to develop an automated inspection solution to address challenges in inspecting railway overhead line equipment within the railway industry. By integrating drone autopilot technology with deep learning-based object detection, this goal can be achieved. Section 2 reviews prior research, Section 3 describes the research and evaluation methods, Section 4 presents results and discussion, and Section 5 concludes the study.

II. PREVIOUS RESEARCH

Object detection technology has been advancing rapidly, with various learning methods available. One representative object detection method is You Only Look Once (YOLO). The architecture of YOLO is shown in Figure 1[7].

Unlike previous mainstream learning methods that employed region proposal techniques—first generating potential bounding boxes in an image, then running classifiers on these proposed boxes, and subsequently performing complex post-processing steps such as refining bounding boxes, eliminating duplicate detections, and re-scoring boxes based on other objects in the scene—YOLO treats object detection as a single regression problem. It directly learns to map image pixels to bounding box coordinates and class probabilities in one step. This approach offers advantages such as faster and more accurate detection compared to conventional methods, with the ability to detect objects in a single glance.

The architecture of Tiny-YOLO, a simplified version of YOLO, is shown in Figure 2[8]. Tiny-YOLO, compared to YOLO, features a simpler structure, which results in lower detection accuracy but faster inference speed. Its high inference speed makes it well-suited for real-time detection. Additionally, being a lightweight model, it requires fewer resources, allowing it to operate stably even in resource-constrained environments such as Raspberry Pi.

In prior research, Yolov3 was used to detect railway overhead equipment. A training dataset consisting of 2,400 images of railway overhead equipment was created. Detection was performed on a PC, achieving the detection accuracy shown in Figure 3. When the detection was conducted on the PC, the detection time was 0.077 seconds, whereas on a Raspberry Pi, it took approximately 15 seconds.

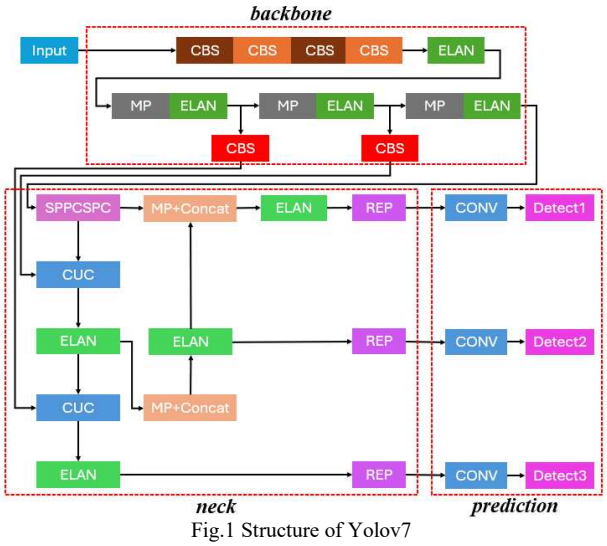

Fig.1 Structure of Yolov7

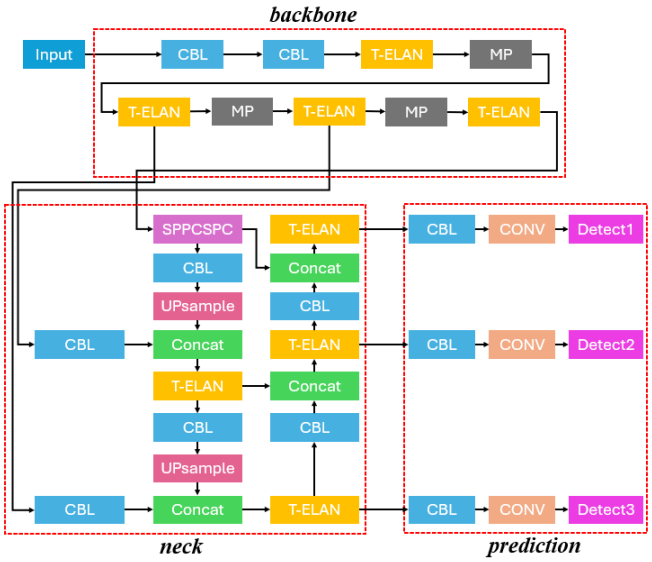

Fig.2 Structure of Yolov7-tiny

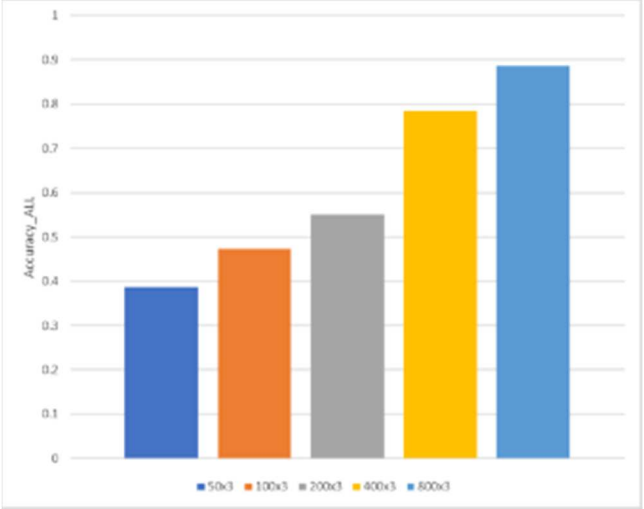

Fig3. Accuracy of train line detection

III. METHOD AND EVALUATION

A. Method

In this study, AI training was conducted using YOLO and Tiny-YOLO version 7 training methods. The resulting models were deployed on a Raspberry Pi for object detection, and their performance was evaluated. The operational environment used for training on a PC included Windows 11 and Python 3.11, while Raspberry Pi 4B ran Python 3.9.

The workflow of this research is shown in Figure 4. First, a dataset was created using 2,400 images of railway overhead equipment. Three types of equipment—connectors, hangers, and insulators—were annotated using the software VOTT. The components and the annotation process are illustrated in Figure 5. The annotations generated by VOTT were in Pascal VOC format (XML files). Since YOLO requires TXT files in YOLO format for training, a conversion was performed. Using the created dataset of 2,400 images, training was conducted for Yolov7 and Yolov7-Tiny on a PC. The training parameters were set as follows: the number of epochs was 300, the batch size was 32, the image size was 640, and the model configuration files used were yolov7.yaml for Yolov7 and yolov7-tiny.yaml for Yolov7-Tiny. Epoch number is the number of iterations of a dataset to adjust parameters in

machine learning, and batch size is the number of groups into which the dataset is divided when training. The model, trained in this way, was then used for evaluation. For the evaluation, a separate annotated test dataset, different from the one used for training, was prepared. The evaluation was carried out by running a dedicated test script. For object detection of railway overhead equipment, the best-performing model during training, best.pt, which achieved the highest validation performance, was utilized

. Next, object detection for railway overhead equipment was performed on a Raspberry Pi, an edge device, using the model trained on a PC. While attempting to run the pretrained model on the Raspberry Pi with the PyTorch framework, errors occurred. To address this, the pretrained model was converted into the ONNX framework format. ONNX, which stands for Open Neural Network Exchange, is a format designed to represent artificial intelligence models such as deep learning and machine learning models. ONNX enables models from various frameworks to be used together and is optimized for hardware, making it suitable for operation on edge devices.

After converting the model to ONNX format, the operational environment was set up on the Raspberry Pi. Object detection was conducted on the Raspberry Pi using the converted model, and measurements were taken for detection time, power consumption, CPU usage, and RAM usage during the process.

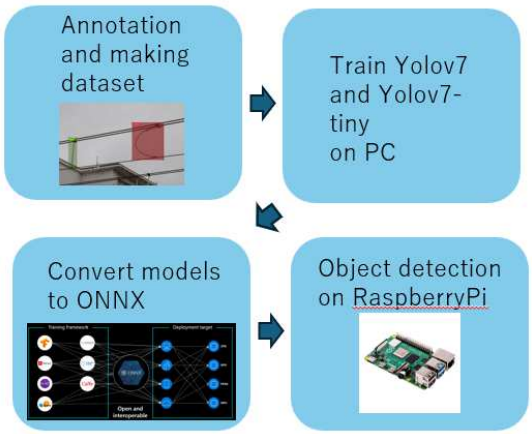

Fig4. Research flow

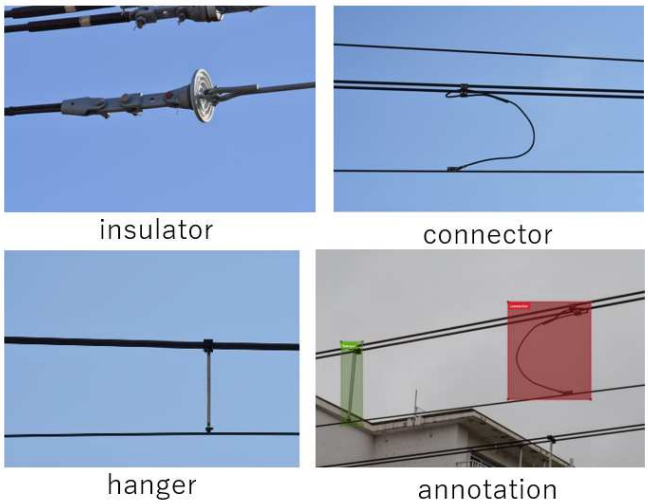

Fig.5 Parts and Annotation

B. Evaluations

When evaluating a trained object detection model, the metric mainly used is mAP and F1-score. To calculate that, four indicators—TP, FP, TN, and FN—are important. True Positive (TP) refers to cases where the object detection model predicts an object and the prediction is correct. False Positive (FP) refers to cases where the model predicts an object, but the prediction is incorrect. True Negative (TN) refers to cases where the model predicts no object and the prediction is correct. False Negative (FN) refers to cases where the model predicts no object, but the prediction is incorrect. Since there are a large number of negative background regions, TN is not very meaningful and is rarely used as a metric. Using the values of TP, FP, and FN, Precision and Recall can be calculated. The formula for Precision and Recall is as follows,

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

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

Precision represents the proportion of correct predictions among all positive predictions made by the model, and it increases as FP decreases. Recall represents the proportion of actual positive samples that the model correctly detects, and it increases as FN decreases.

Because of this, Precision and Recall have a trade-off relationship, meaning that increasing one often results in a decrease in the other. The F1 score is a metric used to evaluate the performance of a classification model, balancing Precision and Recall. When these metrics have a trade-off relationship, the F1 score represents the harmonic mean of Precision and Recall, providing a single value that reflects the overall performance of the model. The formula for calculating the F1-score is as follows,

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

The mAP, which stands for "mean Average Precision," represents the effective average of Precision when plotted with Precision on the vertical axis and Recall on the horizontal axis. The formula for calculating mAP is shown below.

$$AP = \sum_{n=1}^{N}(Recall_n - Recall_{n-1}) \cdot Precision_n \quad (4)$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (5)$$

In this case, the mAP and f1-score calculation will be performed with a confidence score of 0.5. By setting the confidence score to 0.5, it means that any detection with a confidence score of 0.5 or higher will be considered a valid detection. The confidence score is a numerical value that indicates the degree of certainty the model has about the accuracy of a given prediction in object detection and classification tasks. It is typically expressed in the range of 0 to 1, where a score closer to 1 means the model is more confident that the prediction is correct.

The model evaluation is performed on a PC, using a method called k-fold cross-validation. This approach involves using every dataset in the dataset as a test set at least once. A visual representation of k-fold cross-validation is shown in Figure 6. In this experiment, the dataset is divided into five parts, and the evaluation is conducted five times. After the evaluation, the average value of each model's performance is calculated, and the mAP is derived.

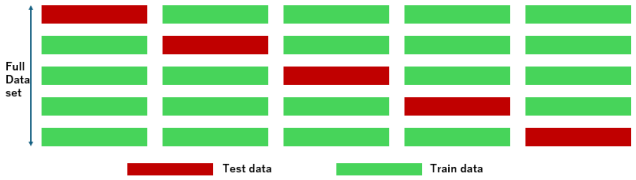

Fig.6 k-fold cross-validation

When measuring CPU and RAM usage on the PC, object detection is performed on 1,000 images consecutively, and the maximum values for both CPU and RAM usage are recorded during this process. For measurements on the Raspberry Pi, object detection is performed on one image for the Yolov7 model and ten images for the Yolov7-Tiny model. The maximum values for CPU usage, RAM usage, and power consumption are then recorded.

IV. RESULT AND CONSIDERATION

In this study, a dataset consisting of 2400 images of railway overhead equipment was created. AI training was conducted using Yolov7 and Yolov7-Tiny, followed by object detection and model evaluation. The object detection results with the trained models are shown in Figure 7. Next, cross-validation was performed to evaluate the models. Both Yolov7 and Yolov7-Tiny were tested by dividing the data into five parts, conducting training and testing five times each. Some of the test results are shown in Figure 8. The average of the five test results was calculated, and the resulting mAP is shown in Figure 9, with the F1-score shown in Figure 10.

The mAP for Yolov7 was 0.988 for hanger, 0.966 for connector, 0.924 for insulator, and 0.959 overall. For Yolov7-Tiny, the mAP was 0.982 for hanger, 0.976 for connector, 0.910 for insulator, and 0.956 overall. The F1-score, calculated for the entire dataset, was 0.96 for Yolov7 and 0.94 for Yolov7-Tiny. Overall, the results show very high scores. Among the components, the insulator had the lowest value, which is likely due to its smaller size compared to the hanger and connector, making it more challenging to detect than the other parts. Although Yolov7-Tiny showed better performance for connectors, Yolov7 outperformed Yolov7-Tiny overall.

The trained models were converted to the ONNX format and object detection was performed on the Raspberry Pi. The CPU and RAM usage, object detection speed, and additional power consumption on the Raspberry Pi were measured. The results are shown in table 1. On the PC, there was no significant difference between Yolov7 and Yolov7-Tiny. However, on the Raspberry Pi, the object detection time using the Yolov7-Tiny model was reduced to approximately one-seventh of that with Yolov7. Additionally, RAM usage decreased to about one-third.

When detecting railway overhead equipment in real-time using a drone, assuming the drone camera's field of view is 84°, a flight speed of 10 m/s, and a distance of 5 m from the railway overhead, the time an object remains fully within the frame is approximately 0.9 seconds. Therefore, it is desirable for the detection time to be less than 0.9 seconds. In the current results, it takes 2.7 seconds to detect one image, meaning that when performing detection to cover the entire railway overhead line, the maximum speed will be about 3.3 m/s. The speed of typical drones varies depending on the model and product, but it ranges from about 8 m/s to 30 m/s. With a movement speed of 3.3 m/s, the drone's performance is not fully utilized for real-time inspection of railway overhead equipment using drones. Considering the drone's flight time and the reduction in inspection time, it is believed that the inference speed needs to be further increased.

Additionally, both the training data and test data used for AI training in this study were captured with a stationary camera. Since no tests were conducted using a drone, there is a potential risk that object detection results might degrade when using a drone-mounted camera. This could be due to image blur caused by the drone's movement or lower image quality from the drone's camera.


Fig.7 Overhead line detection

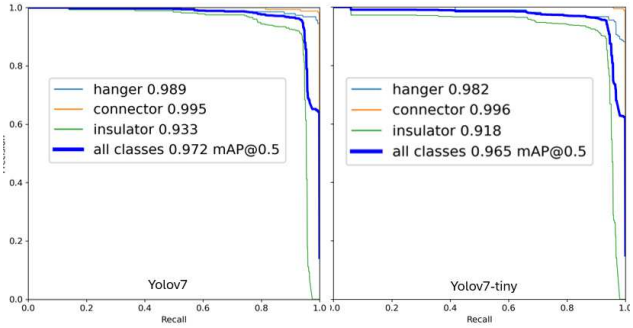

Fig.8 Model evaluation results

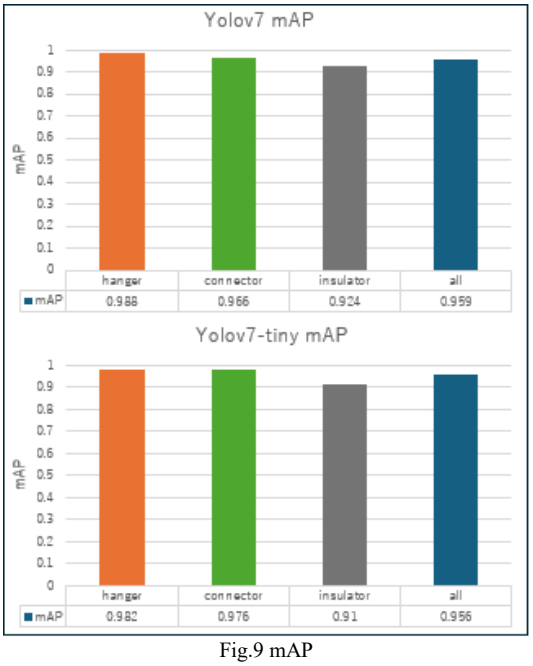

Fig.9 mAP

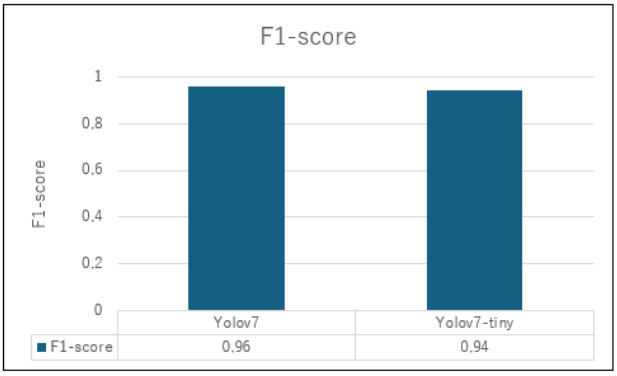

Fig.10 F1-score

TABLE 1 Results for each model

	yolov7 PC	yolov7-tiny PC	yolov7 Raspberry Pi	yolov7-tiny Raspberry Pi
Detection time(s)	0.079	0.065	16.43	2.74
CPU utilization(%)	59.1%	59.3	41	45
RAM consumption(%)	27.1	27.3	38.4	13.8
POWER consumption(W)			4.27	4.21

V. CONCLUSION

Inspection of railway equipment is primarily conducted at night and relies on human visual checks. By leveraging AI and drones, it is expected that labor costs and physical workload can be significantly reduced. As foundational research for future AI implementation, this study aimed to improve inference speed on edge devices while developing an AI model for detecting railway overhead lines. The research involved labeling captured images of railway overhead lines, creating a dataset, and training an object detection AI model using the YOLO method. Object detection was performed on a Raspberry Pi, followed by measurements of CPU and RAM usage, power consumption, and detection time. The trained models achieved mAP and F1-scores of 0.959 and 0.96 for Yolov7, and 0.956 and 0.94 for Yolov7-tiny, respectively. On a PC, no significant differences were observed between Yolov7 and Yolov7-tiny, but on Raspberry Pi, Yolov7-tiny achieved approximately 1/7 the detection time and about 1/3 the RAM usage compared to Yolov7.

Future research should first verify how detection accuracy changes when using a drone. If significant effects are observed, it may be necessary to optimize the model for drone-based detection of railway equipment by including drone-captured images in the training dataset. Additionally, to improve inference speed on edge devices, options include using high-performance edge devices like FPGAs or employing external inference accelerators such as the Coral USB Accelerator or Intel Neural Compute Stick on a Raspberry Pi[9][10]. Alternatively, adopting a more lightweight model than Tiny-Yolo could potentially achieve the inference speed required for real-time detection. Additionally, for the inspection of railway overhead equipment, it is necessary to train AI to detect faults. A significant challenge moving forward is to collect a large amount of data on faulty components, such as detached hangers, corroded connectors, and cracked insulators, and use it for training the AI.

ACKNOWLEDGMENT

The research was conducted as part of the Tokyo City University Prioritized Studies.

REFERENCES

[1] National Institute of Population and Social Security Research. “Population Projections for Japan (2023 revision): 2021 to 2070”. April 26, 2023

[2] Wang J, Fu P, Gao RX, “Machine vision intelligence for product defect inspection based on deep learning and hough transform,” J Manuf Syst, vol. 51, pp. 52-60, 2019.

[3] Kuroda Sadaaki, Akatsuka Yuzo “Study on Japanese Government Assistance for Railway Development in Developing Countries..” Transactions of the Japan Society of Civil Engineers vol 667, pp.15-30, 2001.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”. May 9,2016.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, UC Berkeley, “Rich feature hierarchies for accurate object detection and semantic segmentation”. Oct 22, 2014.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, “SSD: Single Shot MultiBox Detector” Dec 29, 2016.

[7] Bao Rong Chang, Hsiu-Fen Tsai, Chia-Wei Hsieh, “Accelerating the Response of Self-Driving Control by Using Rapid Object Detection and Steering Angle Prediction”. May 23, 2023.

[8] Ryosei Furuichi, Kazuyoshi Takagi, “Using VitisAI FPGA Implementation of YOLOv7-tiny” Proceedings of the DA Symposium 2022,151-156

[9] Marek Kraft, Mateusz Piechocki, Bartosz Ptak, Krzysztof Walas, “Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle” March 2021.

[10] A E Tolmacheva, D A Ogurtsov, M. G. Dorrer, “Justification for choosing a single-board hardware computing platform for a neural network performing image processing” January 2020.