PAPER

Edge Device Verification Techniques for Updated Object Detection AI via Target Object Existence

Akira KITAYAMA[†a)], **Goichi ONO**[†], *and* **Hiroaki ITO**[††], ***Nonmembers***

SUMMARY Edge devices with strict safety and reliability requirements, such as autonomous driving cars, industrial robots, and drones, necessitate software verification on such devices before operation. The human cost and time required for this analysis constitute a barrier in the cycle of software development and updating. In particular, the final verification at the edge device should at least strictly confirm that the updated software is not degraded from the current it. Since the edge device does not have the correct data, it is necessary for a human to judge whether the difference between the updated software and the operating it is due to degradation or improvement. Therefore, this verification is very costly. This paper proposes a novel automated method for efficient verification on edge devices of an object detection AI, which has found practical use in various applications. In the proposed method, a target object existence detector (TOED) (a simple binary classifier) judges whether an object in the recognition target class exists in the region of a prediction difference between the AI's operating and updated versions. Using the results of this TOED judgement and the predicted difference, an automated verification system for the updated AI was constructed. TOED was designed as a simple binary classifier with four convolutional layers, and the accuracy of object existence judgment was evaluated for the difference between the predictions of the YOLOv5 L and X models using the Cityscapes dataset. The results showed judgement with more than 99.5% accuracy and 8.6% over detection, thus indicating that a verification system adopting this method would be more efficient than simple analysis of the prediction differences.

key words: *convolutional neural network (CNN), deep neural network (DNN), object detection, verification using edge devices, automatic verification, embedded ai technique, autonomous driving*

1. Introduction

Software installed in various kinds of devices must be thoroughly verified before such devices are put into operation by customers. The higher the required safety, reliability, and quality of the equipment are, the more important verification becomes, thus increasing the time and cost of verification. For example, vehicle control software supporting automobile systems requires extremely strict verification to ensure the safety and security of drivers, pedestrians, and vehicles [1]–[4]. As illustrated in Fig. 1, when a software defect is found in revision A, the defact must be corrected in revision B. Next, the revision B software must be verified not only in the development environment but also with an experimental vehicle in the real world for final verification to ensure that

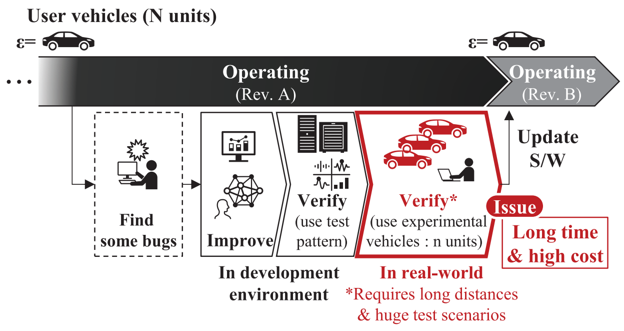


Fig. 1 Software development process for edge devices (e.g., autonomous driving cars).

there is no malfunction. As vehicle manufacturers need to guarantee the safety of their vehicles under all circumstances, the verification using the experimental vehicle will include a wide range of driving situations. If a defect is found during this vehicle verification process, it must be debugged in the development environment and the software fix must be verified again with the experimental vehicle. Thus, each time the software is updated, verification entails a high cost to analyze the results and an enormous amount of time.

One means of reducing the verification cost is to improve the verification quality in the development environment by generating test patterns that reproduce real-world conditions [5]–[7] and selecting efficient verification scenarios [8], [9]. However, it is not realistic to generate an infinite number of possible scenarios, and no matter how efficiently they are verified, the verification process using an experimental vehicle cannot be skipped. This seems to contradict the fact that it is possible to pass verification if the vehicle is tested under a predetermined driving scenario and if it passes with the required driving time and distance. In other words, the two-step process of testing with a sufficiently high-quality simulation and then verifying with an experimental vehicle cannot be skipped.

Furthermore, the application of AI in vehicle control software has rapidly advanced in recent years because of the functional sophistication required for autonomous driving systems. Unlike software written in a rule-based manner, the performance of AI is determined by the training data, training parameters, and learning methods. As a result, software updates often cause performance improvement or degradation that was unintended by the developer. Indeed, researchers have already examined the difficulty and importance of AI verification and how it should be done [10], [11].

Manuscript received October 18, 2023.
Manuscript revised December 1, 2023.
Manuscript publicized December 20, 2023.

[†]Hitachi Ltd. Center for Digital Service Research & Development Group, Sensing Integration Innovation Center, Kokubunji-shi, 185-8601 Japan.
[††]Hitachi Astemo Ltd., Yokohama-shi, 220-0012 Japan.
a) E-mail: akira.kitayama.er@hitachi.com

DOI: 10.1587/transfun.2023EAP1131

For example, consider the case of AI for peripheral recognition, which is essential for many systems, including autonomous driving. It frequently happens that an object that can be recognized by an operating AI version cannot be recognized by an updated version. Accordingly, the AI must be verified very carefully, and we can easily see that the cycle of finding, fixing, and updating bugs occurs more frequently in AI than in rule-based software. Moreover, as AI-capable autonomous driving systems become more widespread, the cost of verifying updated AI versions by using experimental vehicles will become a more critical issue.

In this paper, we examine how to improve the efficiency of validation with real vehicles during updating of an object detection AI [15]–[19], which is one of the most popular AI applications, along with classification tasks [12]–[14] and segmentation tasks [20]–[22]. Section 2 first describes a previously proposed verification method and the challenges in verification with real vehicles, and it then explains our novel method to address these challenges. Then, Sect. 3 describes the effectiveness of the proposed method with experimental results, and Sect. 4 presents our conclusions. This article is based on our conference paper [26].

2. Verification Method with Vehicles

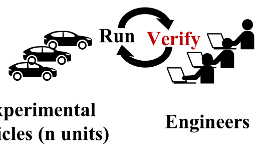
2.1 Verification Method Compared to Driver's Control

In this section, we describe one proposed method to solve the issue of the real-vehicle verification cost. Instead of using several tens of experimental vehicles (n units) as shown in Fig. 2(a), we can increase the efficiency of verification by a factor of N by using vehicles owned by ordinary users (N units, where N≫n), as shown in Fig. 2(b). Implementation of this method requires that verification is performed automatically, without affecting ordinary users. The method illustrated in Fig. 3 was proposed to achieve such automatic verification. The driver's control (via the steering wheel, gas pedal, and brake pedal) and the output of the vehicle control software to be verified are constantly compared. When there is a difference between the control systems, information from the onboard sensors at that time is sent to a server. The software's output is not connected to the car's actuators, and verification is performed only during manual operation. Although this method relies on an assumption that the driver's control is correct, it is much more efficient than verification using experimental vehicles.

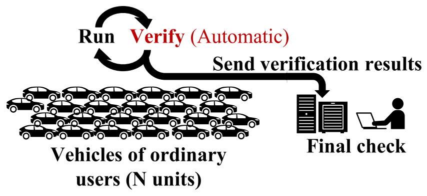
However, because this method has only the driver's control as the correct answer for verification, it is not possible to verify each functional block that constitutes the software or each AI function alone. In addition, because verification can only be performed during manual driving, the coverage and comprehensiveness of the verification scenarios are insufficient. Considering these issues, a possible verification method is to compare the prediction results of the operating AI version (verified) and an updated AI version (the target of verification) during automatic operation, as illustrated in Fig. 4. In this scheme, the operating version has been sufficiently validated and judged to be of practical use. It is then sufficient to show that the updated version does not have degraded performance as compared to the operating version. However, this method has the following issues:

(i) Absence of correct data on edge device:
It is impossible to judge whether the performance difference between the two versions is due to improvement or degradation.

(ii) Comprehensiveness of verification:
The large number of user vehicles running at any given time and location can lead to duplicate or incomplete



(a) Use n experimental vehicles


Run
Verify (Automatic)
Send verification results
Vehicles of ordinary users (N units)
Final check

(b) Use N ordinary user vehicles (N >> n)

Fig. 2 Real-world software verification methods.

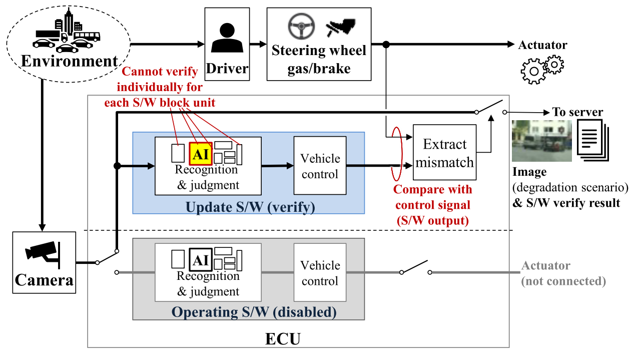


Fig. 3 Software verification with user vehicles.

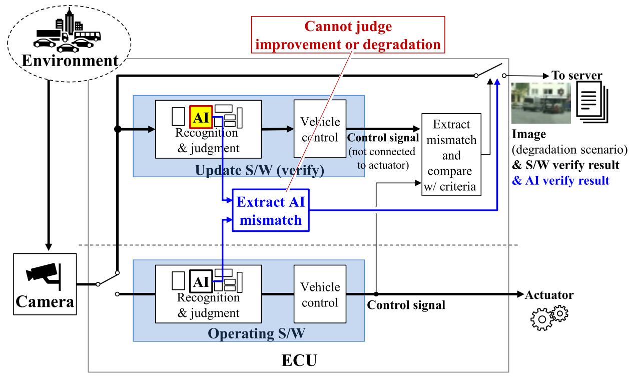

Environment
Cannot judge improvement or degradation
To server
AI
Recognition & judgment
Vehicle control
Update S/W (verify)
Control signal (not connected to actuator)
Extract mismatch and compare w/ criteria
Image (degradation scenario) & S/W verify result & AI verify result
Extract AI mismatch
Camera
AI
Recognition & judgment
Vehicle control
Operating S/W
Control signal
Actuator
ECU

Fig. 4 Software verification by comparing operating and updated software versions.

test scenarios.

(iii) Increased hardware resources:

Efficient implementation is necessary to run both the operating and updated AI versions.

(iv) Management of collected data:

Because scenarios that have been determined to exhibit degradation for a large number of user vehicles are detected and sent to the server, it is necessary to establish a policy on how to handle the large amount of data sent to the server.

In this study, we investigated a new method to ensure the correctness of prediction differences with only the vehicle system in issue (i), which is the most significant issue in verification on a real vehicle.

2.2 Proposed AI Verification Method

To verify the updated AI by comparing its prediction results with those of the operating AI, it is necessary to judge the validity of prediction differences in the absence of correct data. As this requires the performance of the judgment method to exceed that of the AI to be verified, the general idea is that a higher-performance AI should be installed in an edge device. However, such high-performance AIs are difficult to implement in edge devices because of the very high computational cost. Moreover, if a high-performance AI could be implemented in the first place, it could be used without verification.

In contrast, we propose a novel method for verification in a simpler, easier configuration [26]. First, note that the prediction result of an object detection AI comprises the coordinates (region in an image) and class information to identify an object's position. Then, improvement or degradation can be determined according to whether or not an object of the class to be recognized appears in an image cropped from the region of a prediction difference. The key point of the proposed method is that it can enable verification with a very simple structure by simplifying the "object detection task," which predicts a recognition target's position and class from an image, to a "binary classification task," which simply predicts whether an object in a cropped image is a recognition target. On the other hand, this method has an issue of being unable to detect degradation when both the operating and updated AI versions are wrong (missed or false positives). We will need to continue examining this issue, but for the purpose of this paper, we define vehicle verification as follows:

- ✓ In the first phase of vehicle verification, the vehicle is fully verified in the development environment.
- ✓ The final phase of vehicle verification ensures that there are no serious degradations from safe driving.

Hence, to address this issue, the following discussion assumes that in-vehicle verification is valid as long as there is no performance degradation relative to the operating version.

Figure 5 illustrates the proposed method's structure. The performance improvement or degradation by the updated AI is determined according to the decision rules listed in Table 1, which use information on whether a detected target object is captured in the region of the prediction difference in Fig. 5(e) and information on whether the prediction difference was detected by the updated AI (f). For example,

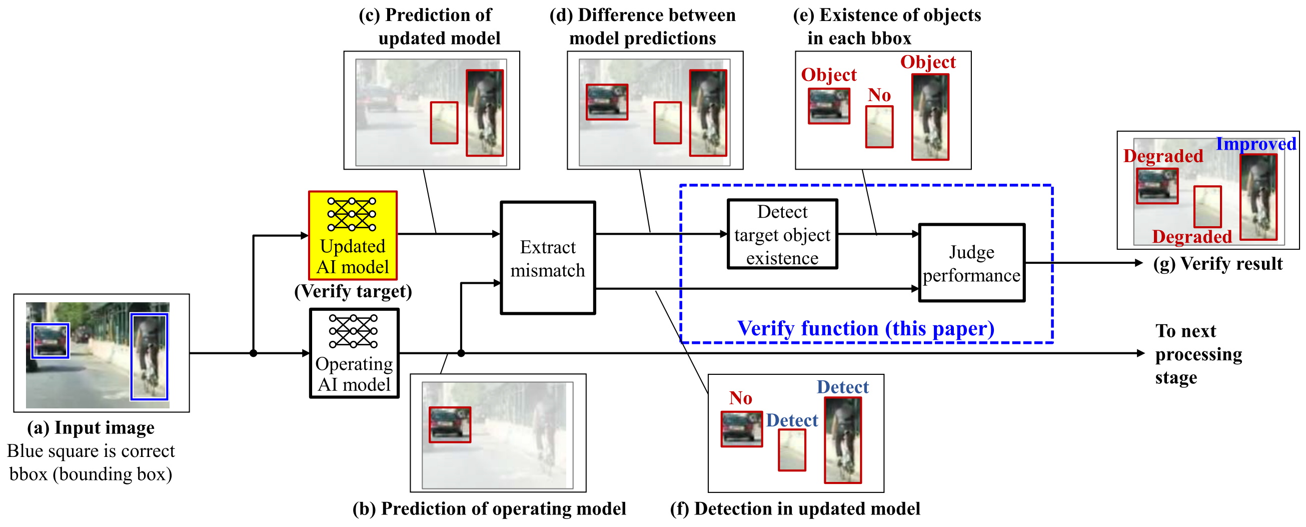

(c) Prediction of updated model
(d) Difference between model predictions
(e) Existence of objects in each bbox
Object
Object
No
Degraded
Improved
Degraded
Updated AI model
(Verify target)
Extract mismatch
Detect target object existence
Judge performance
(g) Verify result
Verify function (this paper)
Operating AI model
To next processing stage
(a) Input image
Blue square is correct bbox (bounding box)
No
Detect
Detect
(b) Prediction of operating model
(f) Detection in updated model

Fig. 5 Proposed verification method, illustrated by a simple, clear example scenario.

Table 1 Rules for judging improvement or degradation by an updated AI.

	Existence of object in bbox	
	Object	None
Detected by new version	Improvement	Degradation
Not detected by new version	Degradation	Improvement

if the updated AI version does not detect a car (c), but the operating version does (b), and if the detected object exists in the region of the prediction difference (e), then the updated AI is degrading the performance (g).

Below, we describe the target performance of the binary classifier, which determines the proposed method's verification accuracy. Because data for verification is continuously input at several tens of frames per second (fps), we assume that the validity of the verification results is ensured by using the continuity and fluctuation of the results over several frames. For example, if a binary classifier with a 95% judgment performance produces the same judgment result for three consecutive frames, the accuracy is 99.9%. This is a probabilistic concept that does not hold true under all conditions; however, in this study, we set a temporary target of 95% for the binary classifier's judgment accuracy.

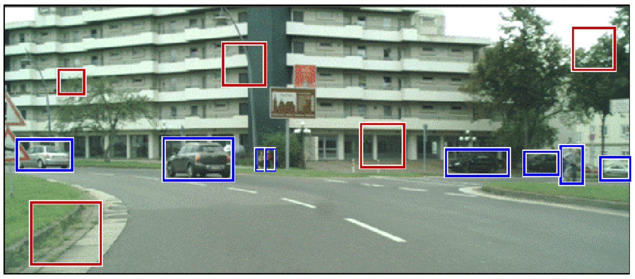
3. Implementation and Evaluation Results

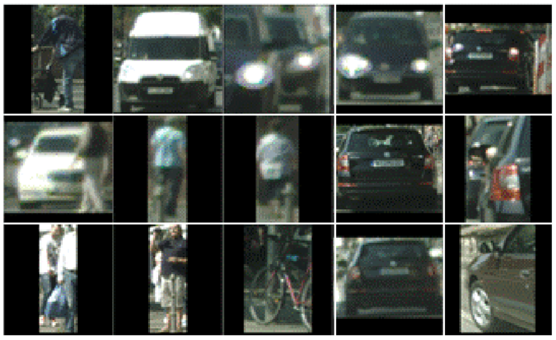
3.1 Design and Evaluation of Binary Classifiers

Here, we describe the binary classifier's design in terms of the training dataset and architecture. The training dataset comprised images in which objects for detection by the object detection AI to be verified were labeled with either "object," indicating existence, or "none," indicating no object. As shown in Fig. 6(a), blue rectangular regions with correct labels (i.e., objects to be detected, such as cars, bicycles, and pedestrians) were cut out and labeled with "object," yielding the results shown in Fig. 6(b). In contrast, the red rectangular regions (Fig. 6(a)) containing no object to be detected (i.e., the background, road surface, buildings, trees, and signs) were randomly cut out and labeled with "none," yielding the results shown in Fig. 6(c). Because of the constraint of a fixed size for input images to the binary classifier, cut-out regions were resized to fit in a 64 × 64 pixels region, and excess areas were filled with zeros.

The training and evaluation datasets for the binary classifiers contained 3,000 and 500 images from the Cityscapes training and evaluation datasets, respectively. The training objects comprised the three classes of vehicles, bicycles, and pedestrians. As listed in Table 2, the images with existing objects included 5,000 extracted labels for training and 2,000 for evaluation, while the images without objects included 10,000 instances of no object for training and 2,000 for evaluation. For convenience, we refer to these datasets collectively as the "object existence dataset" below. Note that if a collected image's size is too small, or if only a small portion of an object is shown, even the AI to be verified may have difficulty detecting the correct label, and it thus may not correctly predict differences. Accordingly, collected images with an area of 1,024 pixel^2 (equivalent to 32×32 pixels) or less were excluded from the dataset.

Next, we describe the configuration of the binary classifier used in this study. It consisted of a simple convolutional neural network (CNN) to suppress the additional computational cost. As shown in Fig. 7, the input was a 64×64 RGB


(a) Example of cropping regions labeled with "object" (blue squares, corresponding to correct labels in the dataset) and "none" (red squares, randomly selected so as not to overlap with correct answer labels) from an image.


(b) Examples of cropped "object" images (objects of the target class for detection by the object detection AI, e.g., cars, bicycles, pedestrians).

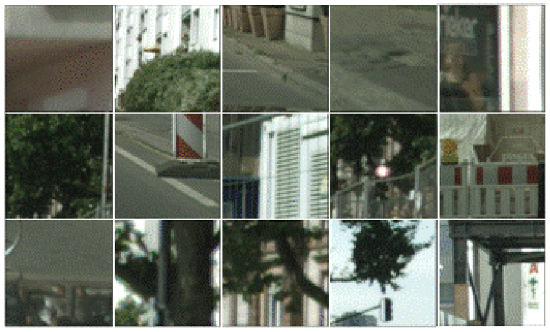

(c) Examples of cropped "none" images (e.g., background, road surface)

Fig. 6 Examples from the training and validation datasets for the TOED.

Table 2 Details of object existence in the datasets for the TOED.

	"Object" images	"None" images	Source
Training dataset	5,000	10,000	Cityscapes training data (3,000 images)
Validation dataset	2,000	2,000	Cityscapes validation data (500 images)

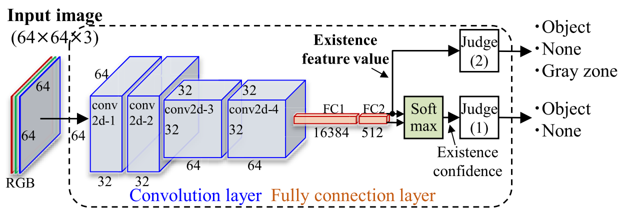
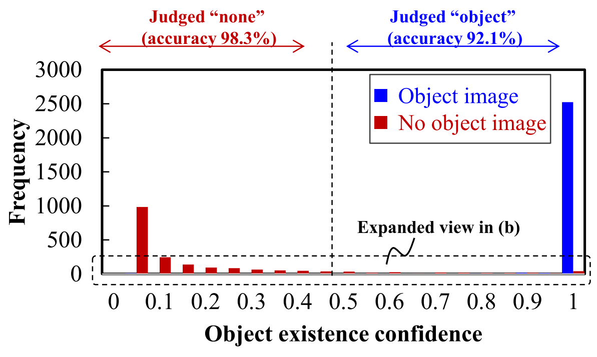
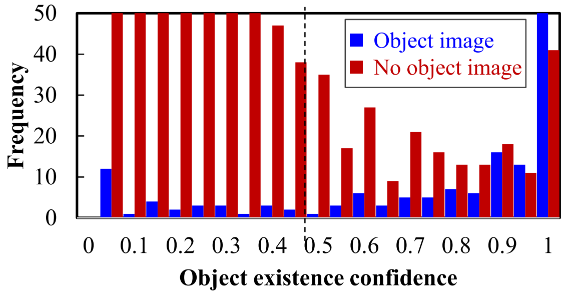


Fig. 7 Configuration of the target object existence detector (TOED).



(a) Histogram



(b) View with expanded vertical axis

Fig. 8 Object existence judgment results in terms of the confidence values for arbitrary cropped images.

image, and the classifier predicted whether an object existed in the image according to classification confidence values. These values were calculated through processing with four convolution layers, two fully connected layers, and softmax activation. Hereafter, we refer to this binary classifier as the "target object existence detector (TOED)."

The multiply-accumulate (MAC) operation in the convolutional and fully connected layers entails 0.5 GFLOPs, which is less than 0.05% of the scale of operations for object detection AI models such as SSD500 [17] and YOLOv5 [19]. When these AI models are implemented in edge devices, it is common to remove 80–90% or more of the operations via model compression techniques [23]–[25]. Even taking this into account, the proposed TOED can verify a target AI with an additional operating cost of 0.5% or less.

Figure 8(a) shows a histogram of the TOED's performance on our object existence dataset. The horizontal axis represents the confidence value indicating the object existence probability output by the detector. When the confidence threshold that an input image had an object was 50% for "Judge (1)" in Fig. 7, the judgment performance A_{total} was 98.8%. Specifically, we define the judgment performance via the following equations:

$$I_{obj} = J_{oo} + J_{on}, \tag{1}$$

$$I_{none} = J_{nn} + J_{no}, \tag{2}$$

$$A_{obj} = \frac{J_{oo}}{J_{oo} + J_{no}} \times 100, \tag{3}$$

$$A_{none} = \frac{J_{nn}}{J_{nn} + J_{on}} \times 100, \tag{4}$$

$$A_{total} = \frac{J_{oo} + J_{nn}}{I_{obj} + I_{none}} \times 100. \tag{5}$$

Here, I_{obj} and I_{none} denote the respective numbers of images with and without objects, and J_{oo}, J_{on}, J_{no}, and J_{nn} denote the respective numbers of object images judged as "object," object images judged as "none," no-object images judged as "object," and no-object images judged as "none." In addition, A_{obj}, A_{none}, and A_{total} denote the respective accuracies of judging "object," judging "none,", and judging the total.

The expanded histogram in Fig. 8(b) shows that many

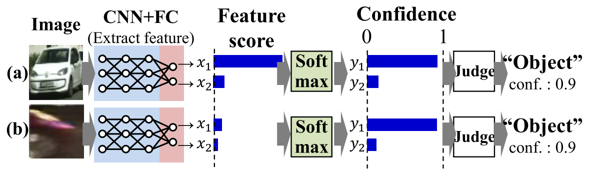


Fig. 9 Feature scores and confidence values for prediction uncertainty: (a) the input image obviously includes a car; (b) the input image is unclear.

cases were judged incorrectly despite high confidence (i.e., high probability). One reason is that the feature values (x_1, x_2) output by all the combined layers in the classifier's final stage were compressed by the softmax activation to a prediction confidence (y_1, y_2) between 0 and 1, thus losing the scalar values of the features. The softmax activation was defined by the following equations, where x_1 denotes the "object existence score," and y_1 denotes the "object existence confidence," as represented by the x-axis in Fig. 8:

$$\begin{cases} y_1 = \dfrac{\exp(x_1)}{\exp(x_1) + \exp(x_2)} \\ y_2 = \dfrac{\exp(x_2)}{\exp(x_1) + \exp(x_2)} \end{cases}. \tag{6}$$

In the example shown in Fig. 9, the x_1 of the extracted feature is high because the image in (a) obviously shows a vehicle, and after softmax processing, the prediction confidence $y_1 = 0.9$. On the other hand, the image in (b) has an indistinct background but also looks like a car. The feature extracted from this image has low values for both x_1 and x_2, but the softmax processing may give the same confidence as in (a), depending on the ratio of these values. In other words, the prediction results are highly uncertain.

Hence, to improve the judgment performance by using the object existence score, we adopted a policy of at least not allowing degradation to be missed. For this purpose, uncertain predictions with small scores, such as the scenario in Fig. 9(b), are judged as belonging to a "gray zone" and are sent to the server in the same way as scenarios that are judged to have degraded performance. Figure 10 shows a histogram of the object existence scores obtained with this approach. For feature values of −3 or lower, A_(none) was 99.0%, and for feature values of 2 or higher, A_(exist) was 99.4%. On

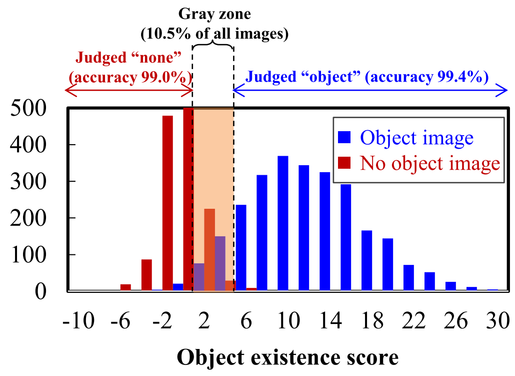


Fig. 10 Object existence judgment results using scores for arbitrary cropped images.

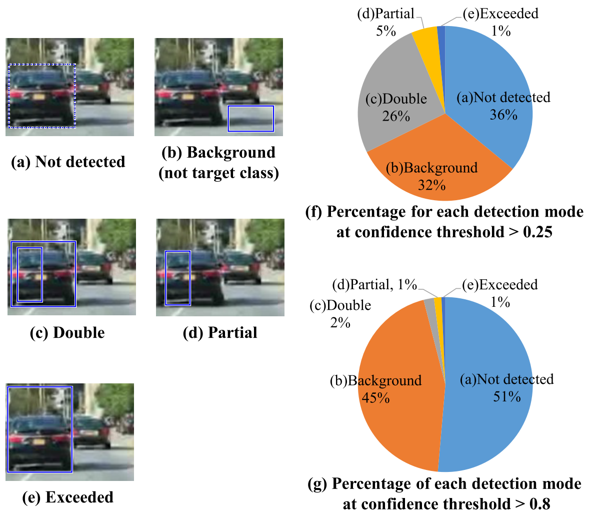


Fig. 11 Examples of each prediction difference mode and their percentages.

the other hand, the interval from −3 to 2 accounted for 10.5% of the total and comprised a mixture of actual objects and nonexistent objects, as given by the following gray zone ratio R_{gz}:

$$R_{gz} = \frac{J_{gz}}{I_{obj} + I_{none}} \times 100. \tag{7}$$

Here, J_{gz} denotes the number of images judged as "gray zone," meaning that the TOED predicted either "object," "none," or "gray zone." In other words, this feature range constitutes a gray zone in which an object's existence cannot be determined. However, by sending such data to the server with a suspicion of degradation to avoid missing objects, we confirmed degradation detection with a judgment accuracy A_{total} of more than 99.2% (with a 10.5% gray zone) for this dataset.

3.2 Evaluation Results of Proposed Verification Method

The performance of the proposed method was evaluated using the configuration shown in Fig. 5. The operating AI version was the L model of YOLOv5 [18], the updated AI version was the X model of YOLOv5, and the input images were 500 evaluation images from Cityscapes. The Intersection over Union (IoU) between bounding boxes (bboxes) in each version's prediction results was calculated for all combinations, and if no combination had a value greater than 0.5, that bbox was considered to be the prediction difference. However, using the rules listed in Table 1 to verify the detection patterns shown in Fig. 11 would lead to errors in cases of (c) double detection, (d) partial detection, (e) exceeded detection including both the background and object, and so on. For example, consider the case where the prediction result of the operating AI is shown in Fig. 11(c) (the outer bbox is correct and the inner bbox is wrong), and the updated AI only predicted the outer bbox. On the other hand, let us consider the case where a operating AI detects both outer and inner bboxes. Therefore, the difference in prediction between the updated and operating AI is inner bbox. In this case, the updated AI does not detect the inner bbox, so we can say that its performance has improved. On the other

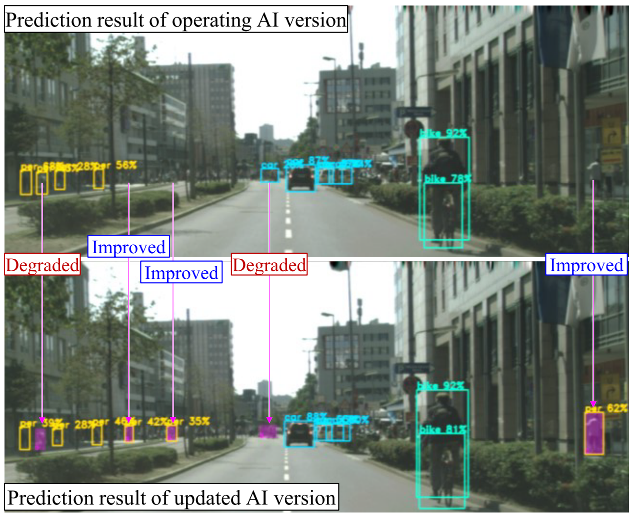

Prediction result of operating AI version
Degraded
Improved
Improved
Degraded
Improved
Prediction result of updated AI version

Fig. 12 Example of differences between the two AI versions' predictions.

hand, since the inner bbox shows only a part of the vehicle, it is judged as an "object" with high probability. Therefore, the performance of the updated AI, which does not detect the inner bbox, is judged "degraded" according to the rules in Table 1. To reduce the occurrence of these complex detection patterns, only results with a predicted confidence of 0.8 or higher were considered in this paper. Figure 11(f) and (g) show the percentages for each detection mode when the confidence threshold for extracting prediction differences was set to 0.8 and 0.25, respectively. Although measures for cases (c–e) are necessary, depending on how strictly they are verified, this paper focuses on verifying cases (a) and (b), which occur the most frequently.

Next, Fig. 12 shows an example of prediction results obtained by the operating and updated AI versions and their prediction differences. In the lower image, the areas of prediction differences are filled in pink. The prediction differ-

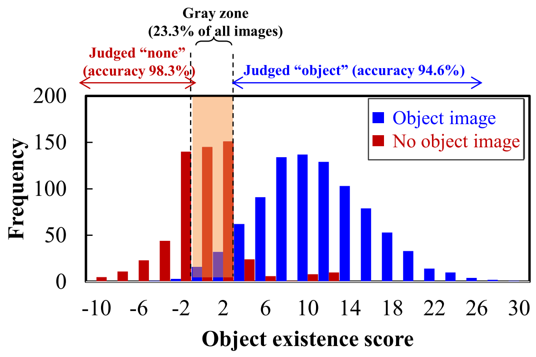


Fig. 13 Judgment results for object existence using the scores for prediction differences between the two AI versions.

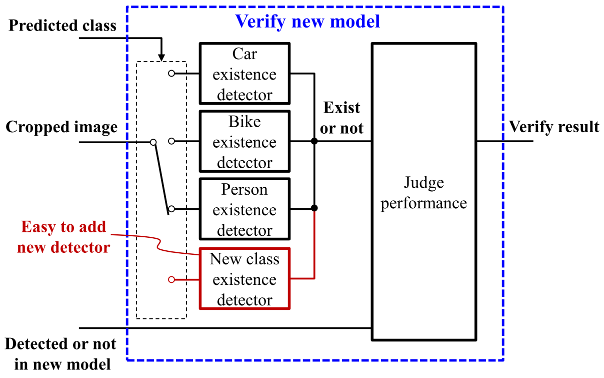


Fig. 14 Configuration with a TOED implemented for each class.

ences with a frame are bboxes predicted only by the updated AI, while those without a frame are bboxes predicted only by the operating AI.

Figure 13 shows the prediction results of the binary classifier described in Sect. 3.1 when the confidence threshold for extracting prediction differences was 0.8. The judgment accuracy was 94.6% for images with objects and 98.3% for images without objects. These results demonstrate the prospect of achieving a total judgment accuracy of more than 90%, but they also clearly revealed the issue of a large number of images in the gray zone (23.3%). One reason for the worse results than those shown in Fig. 10 is that images yielding prediction differences are generally more difficult to recognize, so the binary classifier tends to make wrong decisions. Furthermore, simply using the TOED output (exist/none) is not sufficient to verify class errors in the prediction results. To address this problem, it is not enough to simply review the training dataset; it is necessary to change the configuration of the binary classifier. The next section describes methods for improving verification performance.

3.3 Improved Performance of TOED

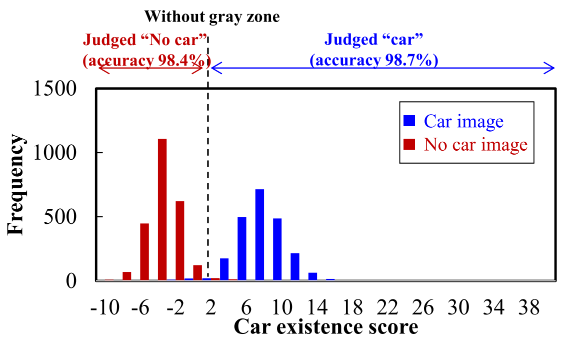
Lastly, we describe the results of our study on an example of classification performance improvement. Currently, the binary classifier's task is to determine whether an object to be verified exists. Therefore, while this method can verify the existence of bboxes, it is not sufficient for verifying the classification results. In addition, the more classes there are to be verified, the higher the performance requirements and the larger the required network size, because the binary classifier needs to learn the features of various objects. Moreover, the addition of new classes when updating the AI version requires retraining and redesign of the binary classifier, which poses a challenge in practical terms. To address these issues while improving the classification performance, we propose a new method that implements a TOED for each class, as shown in Fig. 14, and selects the detector according to the class information of the prediction results. This method has the advantage that the decision performance of classifiers for other classes is not affected when a new class is added, because it is only necessary to design and add a TOED for each class. In addition, the method in Figure 14 can also verify the validity of the estimated classification. For example, when the prediction for "vehicle" is "bus," the TOED for "bus" can judge whether the prediction is a bus or not. This means the class of the prediction can be verified.

With this new method, the structure of the CNN for judgment was the same as that shown in Fig. 7. For example, the training data for the car existence detector used the cropped image of a car and other images (bicycles, pedestrians, background, etc.). For the bicycle and pedestrian existence detectors, we also used the image data of the respective class and the image data of other objects for training. The evaluation results for each detector's performance are shown in Fig. 15. The percentages of gray zones that were difficult to judge were 0%, 9.5%, and 7.8% for cars, bicycles, and pedestrians, respectively. The total percentage of gray zones decreased from 23.3% to 8.6%, thus confirming the advantage of implementing a detector for each class.

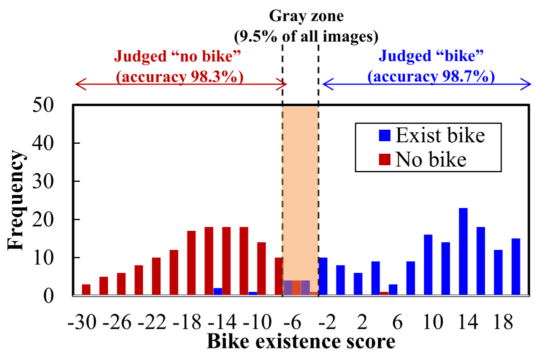
As shown in Fig. 15, the judgement accuracy is more than 98%, which appears to be a good result. However, the accuracy should be improved as much as possible, since a 1~2% performance degradation will be missed. As one method to improve the judgment accuracy as much as possible, we propose the configuration and processing flow shown in Fig. 16. This method uses all of the multiple TOEDs installed in Fig. 14 and combines their results.

As shown in the flowchart in Fig. 16(b), if the result of the TOED for the predicted class is "Exist" and any other TOED is judged to be "Exist", the predicted class may be wrong and is judged to be in the "Gray zone". This allows the accuracy of the validation to be improved by finding possible errors among the results that TOED judges to be "Exist". The TOED "Exist" accuracy was 99.8% for cars, 98.1% for bike, and 99.0% for person, respectively, for a total improvement to 99.5%.

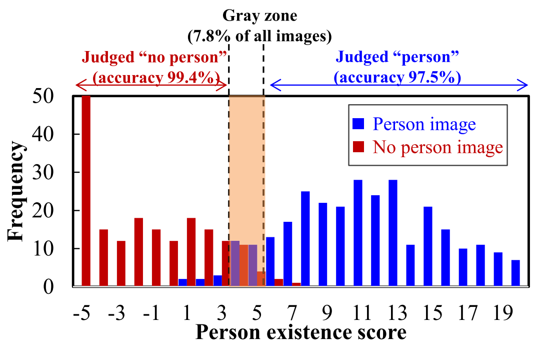
Finally, we compared the verification accuracy of the proposed method (Fig. 5) with the accuracies of the standard verification method of simply comparing the operating and updated AI versions (Fig. 4) and an ensemble method with an additional object detection AI. The ensemble-based object detection AI used three YOLOv5 M models (49.0 GFLOPs/model) [19] trained with random initial pa-



(a) Car existence score histogram

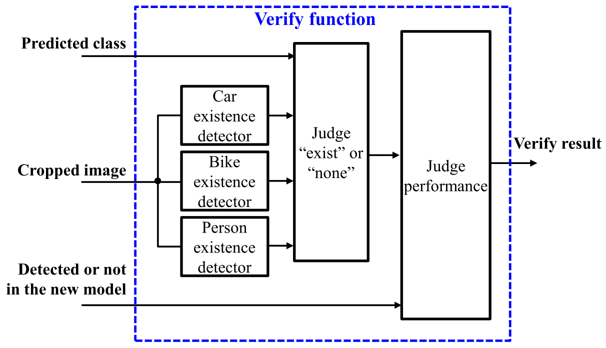


(b) Bike existence score histogram

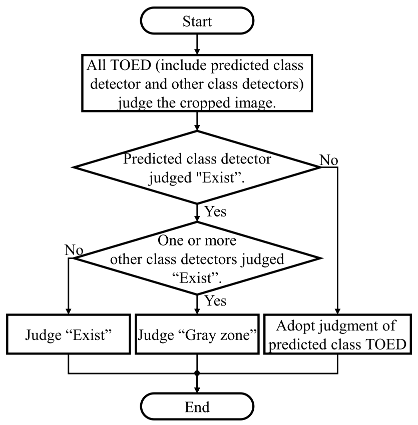


(c) Person existence score histogram

Fig. 15 Judgment results with a TOED implemented for each class, in terms of the scores for prediction differences between the two AI versions.



(a) Configuration


Start
All TOED (include predicted class detector and other class detectors) judge the cropped image.
Predicted class detector judged "Exist".
No
Yes
One or more other class detectors judged "Exist".
No
Yes
Judge "Exist"
Judge "Gray zone"
Adopt judgment of predicted class TOED
End

(b) Flowchart

Fig. 16 Method for improving judgement accuracy of verify function using multiple TOEDs.

Table 3 Degradation detection accuracy for each verification method with updated software.

Automatic verification method for AI	Verification accuracy [%]	Ratio of gray zone [%]	Additional operation cost [GFLOPs]
Simple comparison of operating and updated AIs (Fig. 4)	50.0 (Cannot judge)	---	---
Majority voting on ensemble object detection AI results	99.2	---	147.0*
TOED (this paper, Fig. 5)	**96.5**	**23.3**	**0.5****
Separate TOEDs for each class (this paper, Fig.14)	**98.6**	**8.4**	**0.5****
Use all separate TOEDs (this paper, Fig. 16)	**99.3**	**8.4**	**1.5*****

* Three YOLOv5 M models in parallel.
** Four convolution layers and two fully connected layers.
*** Three TOEDs were implemented, one for each class.

rameters on the Cityscapes dataset. Majority voting on the three model's inference results was applied to detect degradation in the updated AI version. Table 3 lists the verification accuracy and its breakdown for each method, as well as the cost of additional operations. The results confirm that this method can achieve high verification accuracy with very little additional operation cost.

4. Conclusion

In this paper, we have proposed a novel verification method for an object detection AI, which is the most popular AI in the field of autonomous driving, to reduce the cost of verifying an updated AI version in edge devices such as vehicle ECUs. The method uses a target object existence detector (TOED) consisting of a binary classifier that compares the prediction results of the operating and updated AI versions to judge whether there is an object to be detected in the region where a prediction difference occurs. Using the results of this TOED judgement and the predicted difference, an automated verification system for the updated AI was constructed. As a result, performance improvement or degradation in the updated AI version can be recognized even in edge devices with no correct data. The TOED was used to determine the prediction differences between the L and X models of YOLOv5 on the Cityscapes dataset, and the resulting accuracy was more than 96.5%. In addition, by implementing a TOED for each class, the percentage of "gray zone" cases was reduced from 22.3% to 8.6%, and the judgement accuracy to 98.6%. It was also confirmed that using all the TOED results introduced for each class improved the judgment accuracy to 99.5%, thus confirming the validation method's practicality. In the future, we aim to further improve the TOED's accuracy and increase its efficiency of using hardware resources. We also plan to investigate a system that guarantees the comprehensiveness of verification, with the goal of putting this verification technology to practical use.

Acknowledgments

We thank K. Osada, Y. Asada, and H. Onozuka Hitachi Astemo, Ltd., for providing insights on the requirements and technology trends related to autonomous driving systems. We would also like to thank Professor M. Nagata of University Kobe for his guidance in writing this paper.

References

[1] P. Koopman and M. Wagner, "Toward a framework for highly automated vehicle safety validation," Proc. WCX World Congress Experience, 2018.
[2] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," IEEE Access, vol. 8, pp.87456–87477, May 2020.
[3] C. King, L. Ries, J. Langner, and E. Sax, "A taxonomy and survey on validation approaches for automated driving systems," 2020 IEEE International Symposium on Systems Engineering (ISSE), 2020.
[4] C. Neurohr, L. Westhofen, M. Butz, M.H. Bollmann, U. Eberle, and R. Galbas, "Criticality analysis for the verification and validation of automated vehicles," IEEE Access, vol.9, pp.18016–18041, Jan. 2021.
[5] S.S. Shadrin, A.M. Ivanov, and D.A. Makarova, "Methods of parameter verification and scenario generation during virtual testing of highly automated and autonomous vehicles," 2022 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED), 2022.
[6] C.M. Berumen and M.I. Akbaş, "Abstract simulation scenario generation for autonomous vehicle verification," 2019 SoutheastCon, 2019.
[7] H. Hsiang, K. Chen, and Y. Chen, "Development of simulation-based testing scenario generator for robustness verification of autonomous vehicles," 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET), 2022.
[8] M. Mittelsteadt, "AI Verification," Mechanisms to Ensure AI Arms Control Compliance 2021, 2021.
[9] CSET Issue Brief, https://cset.georgetown.edu/wp-content/uploads/AI_Verification.pdf (online)
[10] D. Åsljung, J. Nilsson, and J. Fredriksson, "Using extreme value theory for vehicle level safety validation and implications for autonomous vehicles," IEEE Trans. Intell. Veh., vol. 2, no.4, pp.288–297, Dec. 2017.
[11] L. Klitzke, C. Koch, A. Haja, and F. Koster, "Real-world test drive vehicle data management system for validation of automated driving systems," VEHITS 2019 - 5th International Conference on Vehicle Technology and Intelligent Transport Systems, pp.171–180, 2019.
[12] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Proc. Adv. Neural Inf. Process. Syst., vol.25, pp.1097–1105, 2012.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," CVPR, 2015.
[15] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," NIPS, 2015.
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. IEEE Comput. Vis. Pattern Recognit., June 2016.
[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A.C. Berg, "SSD: Single shot multibox detector," ECCV2016, pp.21–37, 2016.
[18] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018.
[19] G. Jocher, "YOLOv5," 2020, [online] Available: https://github.com/ultralytics/yolov5
[20] T. Sugirtha and M. Sridevi, "Semantic segmentation using modified U-Net for autonomous driving," 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2022.
[21] R. Anantharaman, M. Velazquez, and Y. Lee, "Utilizing mask R-CNN for detection and segmentation of oral diseases," 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp.2197–2204, 2018.
[22] C. Liu, H. Gao, and A. Chen, "A real-time semantic segmentation algorithm based on improved lightweight network," 2020 International Symposium on Autonomous Systems (ISAS), 2020.
[23] S. Han, H. Mao, and W.J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2016, https://arxiv.org/abs/1510.00149 (online).
[24] J.H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," CoRR, abs/1706.05791, 2017.
[25] D. Murata, T. Motoya, and H. Ito, "Automatic CNN compression system for autonomous driving," 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019.
[26] A. Kitayama, G. Ono, and H. Ito, "Edge device verification techniques for updated object detection AI via target object existence detector," 20th IEEE Autonomous and Trusted Vehicles Conference (ATC 2023), 2023.

Akira Kitayama received B.S. and M.S. degrees in engineering from Kyoto Institute of Technology, Kyoto, Japan, in 2005 and 2007. In 2007, he joined the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan, where he has been engaged in research and development of RF circuits, transmission lines that transfer signals with GHz-order frequency, signal processing for automotive mm-wave radars, and embedded AI technique for edge device such as automotive ECU etc. He is a researcher with Hitachi, Ltd., Center for Digital Service Research & Development Group, Sensing Integration Innovation Center, Tokyo, Japan.

Goichi Ono received the B.S. degree in electrical and industrial engineering and the M.S. degree in materials engineering from Hiroshima University, Hiroshima, Japan, in 1996 and 1998, respectively, and received Ph.D. degrees in System informatics from the Kobe University in 2016. In 1998, he joined the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan, where he had been engaged in the research and development of high-speed and low-power CMOS circuit techniques for microprocessors. In the field of wireless systems, from 2003 to 2006, he and his team developed wireless sensor network based on the Ultra-wideband. From 2007, he engaged in the research of high-speed wireline communication technology for 100-Gigabit Ethernet. From 2010 to 2015, he had been working on research of low power SoC in advanced process technology. From 2016, he is working on research of DNN compression techniques.

Hiroaki Ito received B.S. and M.S. degrees in electronic information engineering from Yokohama National University, Kanagawa, Japan, in 1990 and 1992. In 1992, he joined Hitachi Ltd., Tokyo, Japan, where he has been engaged in research and development of image compression and decompression technology, and embedded image quality improvement technology. In 2017, he joined Hitachi Astemo Ltd., Kanagawa, Japan where he has been engaged in development of autonomous driving system.