International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I9P137

Volume 72 Issue 9, 405-413, September 2024
© 2024 Seventh Sense Research Group®

Original Article

Validation of Speed Limit Sign Recognition System in Virtual Environments: A Simulation-Based Approach

Sangjoong Kim[1,2], Dongha Shim[2*]

[1]*SCS, SIEMENS, Korea.*
[2]*Department of MSDE, Seoul National University of Science and Technology, Korea.*

[*]*Corresponding Author: dongha@seoultech.ac.kr*

Received: 13 May 2024 Revised: 08 August 2024 Accepted: 26 August 2024 Published: 28 September 2024

Abstract - *This paper presents a comprehensive approach to validate a speed limit sign recognition system, which is one of the Advanced Driving Assistance Systems in the virtual environment. Its necessity and advantages are emphasized for enhancing automotive safety and efficiency. The recognition of speed limit signs is highlighted as crucial to improving the functionality of driver assistance systems and the development of autonomous vehicles. However, the testing of these recognition systems using actual vehicles can be identified as entailing latent risks in addition to being time-consuming and financially demanding. To address these challenges, a simulation-based validation method is proposed, eliminating the hazards and reducing both the time and financial costs associated with real-world testing. Most of the research has used simulation techniques to test or enhance the performance of the algorithm that comprises the system, but usually neglecting the reliability of the simulation-based validation themselves. This paper focuses on validating the system in simulation and the reliability of the virtual environments through experiments. Finally, the effectiveness of simulation-based validation is elicited. The recognition system used in this paper is based on the You Only Look Once (YOLO) algorithm, renowned for object detection tasks. A diverse set of virtual data to mimic a wide range of real-world scenarios has been used to test the system. This paper presents a detailed comparison between the outcomes derived from tests conducted with real data and those obtained from virtual environment simulations. The results suggest that simulation-based validation can be a possible method for assessing speed limit sign recognition systems, with performance closely matching that in real-world conditions.*

Keywords - *Advanced Driving Assistance System (ADAS), Speed Limit Sign Recognition (SLSR), Object detection, You Only Look Once (YOLO), Digital twin.*

1. Introduction

1.1. Speed Limit Sign Recognition

The advent of autonomous driving technologies and Advanced Driver Assistance Systems (ADAS) has necessitated the accurate recognition of road signs, particularly speed limits, to ensure the safety and legality of vehicle operations. [1] Speed Limit Sign Recognition (SLSR) is one of the crucial functions within ADAS and autonomous vehicles, enabling vehicles to detect and respond to speed limits accurately. By relying on the system, the vehicle can prevent speeding violations and reduce the risk of traffic accidents. However, SLSR performance can be challenged by factors such as driving environment, visibilities, and weather conditions.

For example, a road can have varying regulated speeds in each certain section, and visibility can be affected by complex sign placements. Because of the ability to accurately recognize the speed limit signs in real driving conditions for autonomous vehicles, the validation of the system is also necessary.

1.2. Challenges and Limitations of Simulation-Based Validation

With the increasing complexity of ADAS and autonomous driving systems, traditional validation methods involving real-world vehicle testing face significant challenges, including safety risks, high costs, and extensive time requirements. [2] It means exploring alternative validation approaches that can mitigate these issues while ensuring comprehensive system evaluation. Simulation-based validation emerges as a promising solution, offering the potential to assess the performance of speed limit sign recognition systems safely, efficiently, and cost-effectively. [3] This approach allows for the creation of controlled virtual environments that can simulate a vast array of driving scenarios which are impractical, dangerous, or impossible to replicate in the real world. [4] Despite the widespread use of simulation techniques for algorithm performance testing and enhancement within such systems, the reliability and validity of these simulation-based approaches themselves often remain unexamined. In other words, while simulations are frequently employed to assess how well an algorithm might perform under various conditions, there is often insufficient study on whether these simulations accurately reflect real-world scenarios.

This lack of validation causes concerns about the extent to which results obtained from simulations can be trusted or generalized to real-world applications. Without thorough examination and validation of the simulation environments themselves, there remains a risk that the performance metrics derived from these simulations might not fully capture the true capabilities or limitations of the algorithms in practical settings.

This paper will explain the workflow for evaluating the SLSR system based on simulation, from the data collection stage to the learning and evaluation stages. In addition, it aims to bridge this gap by not only validating a speed limit sign recognition system within a simulated environment but also evaluating the fidelity and reliability of the simulation framework itself. By leveraging the You Only Look Once (YOLO) algorithm [5], known for its efficacy in object detection tasks, this paper conducts a series of experiments using diverse virtual data designed to reflect real-world driving conditions.

A comparative analysis between results obtained from real-world data and those generated within virtual simulations is presented, underscoring the potential of simulation-based validation as an effective method for system assessment. The findings aim to demonstrate that simulation-based approaches can closely mirror real-world performance, thereby validating their utility in the development and testing of autonomous vehicle technologies.

2. Simulation-Based Validation Methodology

For training on custom data, a process shown in Figure 1 is required. The first step of training YOLO with custom data is to prepare a dataset with labeled images, including bounding boxes and class information. [6] The dataset is exported to the code editor, such as Google Colab, Jupyter Notebook, etc. Creating Yet Another Markup Language (YAML) configuration file detailing paths to the training and validation datasets and class names to train the YOLO algorithm tailored to the custom data. [7] Next, the environment required for YOLO, including Python and the necessary libraries, is set up. Then, the training process (Train model step) will be initiated using the prepared data and YAML file. Finally, the trained model's performance is evaluated and tested with a separate dataset to assess its effectiveness (Prediction step). The detailed steps and guidance are presented hereafter.

2.1. Data Preparation

The data preparation process for YOLO involves collecting images of objects to be recognized by the model. Each image must be annotated with bounding boxes around the objects and labels indicating the class of each object. This typically involves using an annotation tool to manually draw boxes and assign labels. The dataset should be diverse, covering various angles, lighting conditions, and backgrounds to improve the model's accuracy and robustness. This prepared dataset is then split into training, validation, and test sets for the model to learn and validate its predictions.

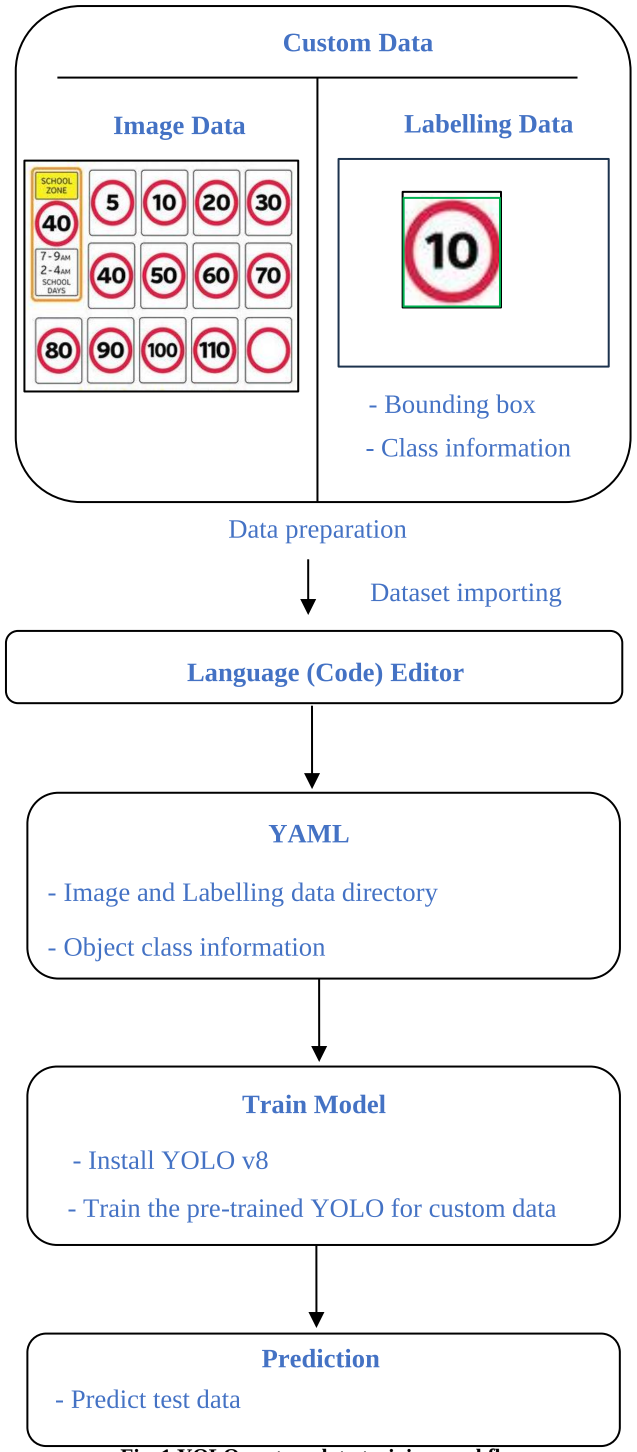


Fig. 1 YOLO custom data training workflow

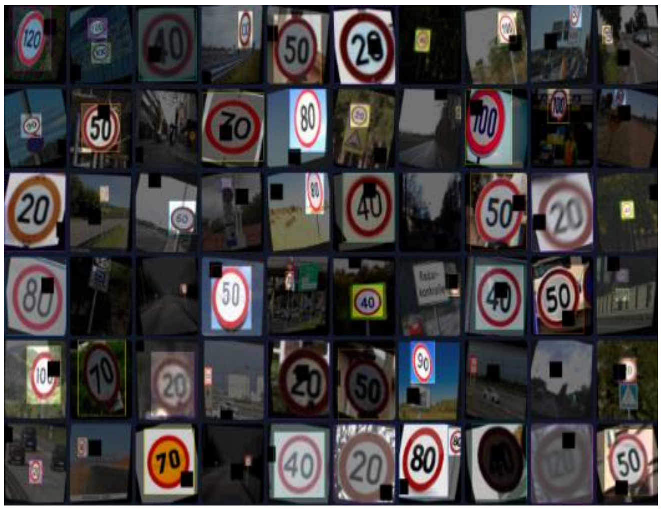


Fig. 2 Speed limit sign dataset images

Table 1. Augmentations of dataset

Techniques	Details
Crop	Minimum Zoom, 20% Maximum Zoom
Rotation	Between -15° and +15°
Shear	±15° Horizontal, ±15° Vertical
Saturation	Between -25% and +25%
Brightness	Between -25% and +25%
Blur	Up to 5.5px
Noise	Up to 5% of pixels
Cutout	1 box with 20% size each

In this paper, the training dataset came from RoboFlow. [8] RoboFlow is a platform designed to help developers, researchers, and companies streamline and enhance their computer vision projects. It provides tools and infrastructure to prepare, annotate, manage, and deploy datasets for machine learning, especially focusing on image recognition tasks.

The training, validation, and test sets have 7419, 510, and 475 images, respectively. Each dataset has 10 classes (20, 30, 40, 50, 60, 70, 80, 90, 100, 120) which means velocity on the speed limit signs. [9] Figure 2 shows some samples of the dataset images.

Table 1 describes various data augmentation techniques for image processing, which can enhance the diversity and robustness of a dataset used for training machine learning models, particularly in computer vision tasks. [10] Techniques such as zooming, rotation, shear, adjustments to saturation and brightness, adding blur and noise, and applying cutout augmentations help models generalize better by simulating a wide range of real-world conditions and variations in the input data.

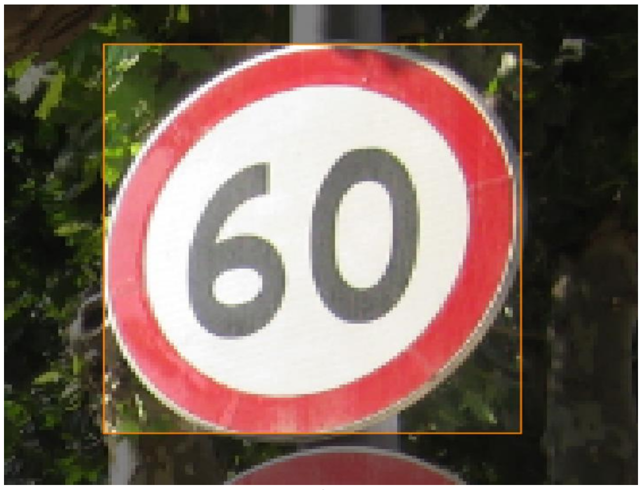


Fig. 3 A sample image of the dataset (60km/h)

Table 2. Bounding box information

	Label	x	y	Width	Height
Value	60	80.06	72.81	108.32	120.50

All the image data are annotated to have information on the bounding box, including label, x, y, width, and height. Figure 3 indicates one of the dataset images, and the orange box represents the bounding box. The information in Table 2 describes an object bounded by a box within the image in Figure 3. They are typically used in machine learning for object detection tasks. "Label" identifies the object, "x" and "y" represent the coordinates of the object's center, while "width" and "height" denote the dimensions of the bounding box surrounding the object. This format helps models like YOLO understand where objects are located in an image and what they represent.

2.2. Custom Data Training by YOLO

YOLO algorithms are designed to divide the image into a grid and predict bounding boxes and class probabilities for each grid cell simultaneously. [5] This approach contrasts with traditional methods that typically scan the image multiple times to detect objects. Before the advent of YOLO, the Faster R-CNN (Regions with Convolutional Neural Networks features) architecture was widely used, but its maximum performance of 7 FPS lacked real-time capabilities, making it impractical. However, the emergence of YOLO in 2015, with its average performance of 45 FPS, marked a revolutionary development in the field of object detection. [5] The YOLO framework has evolved through various iterations, each improving upon the last in terms of detection accuracy, speed, and model complexity. YOLO v8, which is the latest version, is applied to detect the speed limit sign in this paper. Figures 4 and 5 compare different versions of the YOLO object detection models. [11] Figure 4 shows the trade-off between the number of parameters in millions (M) and performance measured by COCO mAP$^{50\_95,}$ which indicates the 'mean Average Precision' from 50% to 95% IoU (Intersection over

Union) thresholds on the validation set. Figure 5 is another plot that depicts the relationship between latency on a combination of hardware and software specifications used for measuring the performance of machine learning models such as A100 TensorRT FP16 and the same performance metric. In both plots, YOLO v8 appears to achieve higher performance with fewer parameters and lower latency compared to its predecessors, indicating improvements in both efficiency and speed of detection. YOLO v8 is a state-of-the-art that has been trained on the COCO (Common Objects in Context) dataset, among others. The COCO dataset provided by Microsoft is a large-scale object detection, segmentation, and captioning dataset that includes hundreds of object categories and millions of images. [12] The training of YOLO on the COCO dataset involves feeding the neural network a large number of images with corresponding labels that identify and localize various objects in the images. Each object in an image is enclosed by bounding boxes with a class label assigned to it, indicating what the object is. Finally, YOLO learns to predict these bounding boxes and class labels directly from full images in a single pass.

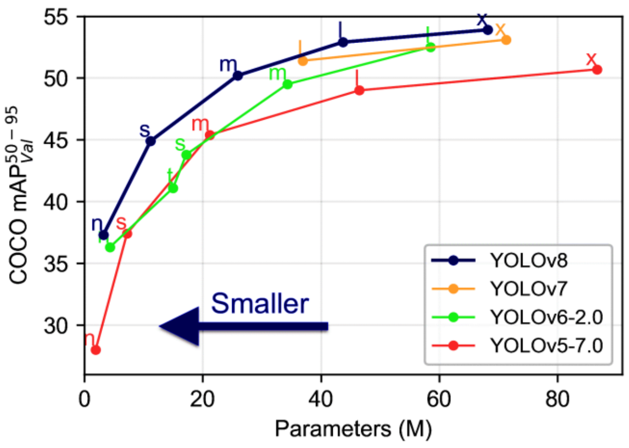


Fig. 4 YOLO v8 performance by the number of parameters [11]

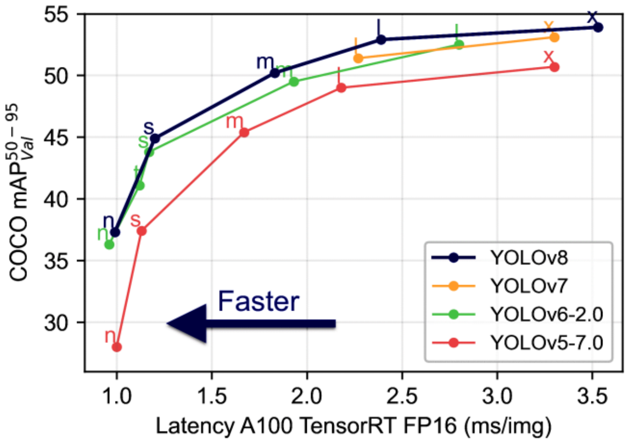


Fig. 5 YOLO v8 performance by latency [11]

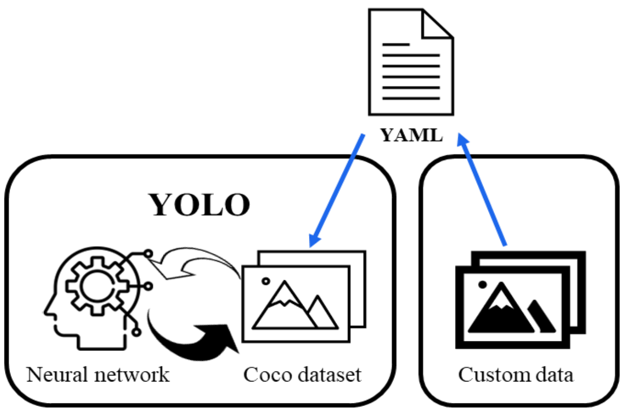


Fig. 6 Custom data training network with YAML

Creating a YAML configuration file for YOLO involves specifying paths to training and validation datasets, listing class names, and possibly adjusting other settings like model architecture or hyperparameters. This file serves as a guide for the YOLO training process, ensuring the model knows where to find data and how to interpret it. The YAML format is chosen for its readability and simplicity, making it easy to edit and understand. [7] Eventually, the custom data training network is shown in Figure 6 because the pre-trained algorithms, which mean the Neural network and the custom data corresponding to the speed limit sign from RoboFlow, will be used for object detection. In other words, the YAML file has a role in delivering the customer data to the learning algorithm. Following the creation of the YAML configuration file and the setup for custom data training with YOLO, the model was implemented and trained using Google Colab. [13] Google Colab offers a cloud-based environment that provides free access to GPUs and TPUs, which significantly accelerates the training process for deep learning models like YOLO. This platform was chosen for its ease of use, accessibility, and ability to handle large datasets effectively without the need for local computational resources.

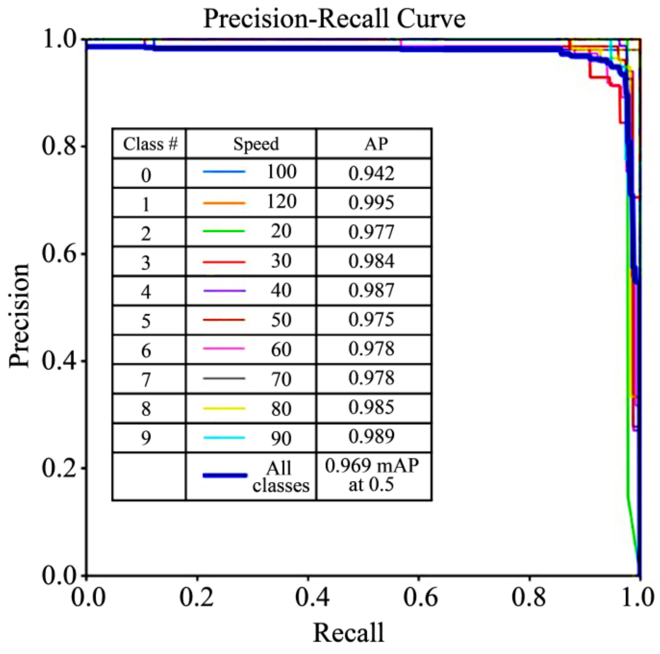
2.3. Trained Model Prediction Performance

Evaluating the performance of a YOLO algorithm for detecting speed limit signs with a Precision-Recall (PR) curve is important for several reasons:

- Balancing Precision and Recall: PR curves illustrate the trade-off between precision (how many selected items are relevant) and recall (how many relevant items are selected). Because both are critical for speed limit sign detection, high precision is required to avoid misidentifying non-speed limit signs as speed limits and high recall is required to ensure all speed limit signs are detected. [14]
- Performance at Various Thresholds: The PR curve shows how the model's performance varies at different threshold settings for classifying an object as a speed limit sign. This helps in selecting an optimal threshold that balances false positives and false negatives according to the specific requirements of the application. [14]

- Model's Ability in Different Conditions: In images, speed limit signs may appear in various conditions and angles. The PR curve can give insights into how well the model generalizes across these different conditions. [15]
- Quantitative Analysis: The Area Under the Curve (AUC) of the PR curve provides a single number to quantify the model's performance, which can be a useful summary statistic when comparing or reporting performance. [16]

In summary, PR curves provide a comprehensive picture of a model's predictive performance and are an essential tool in the evaluation and optimization process for models intended. Figure 7 indicates a PR curve graph with 30 epochs. In the graph, each colored line means the different classes, the speed limits for which the system has been trained to recognize. Each point on the PR curve represents a different threshold of classification probability. A model with perfect classification would have a curve that goes to the top-right corner of the plot, meaning it achieves 100% precision and 100% recall. The legend on the right indicates the Average Precision (AP) score for each speed limit class, with "all classes" showing the mean Average Precision (mAP) across all the speed classes at an IoU threshold of 0.5. (The IoU threshold is a measure of overlap between the predicted bounding box and the ground truth, with 0.5 typically indicating a 50% overlap requirement for a positive prediction.) AP implies the shape of the PR curve, and mAP is the mean of APs across all classes or thresholds, providing a single number to represent the overall performance of the model on the detection task. In this case, most of the AP values are very high (above or near 0.95), and the mAP of 0.969 at an IoU threshold of 0.5 is quite high, indicating that the model performs well across all speed limit classes being evaluated.



Class #	Speed	AP
0	100	0.942
1	120	0.995
2	20	0.977
3	30	0.984
4	40	0.987
5	50	0.975
6	60	0.978
7	70	0.978
8	80	0.985
9	90	0.989
	All classes	0.969 mAP at 0.5

Fig. 7 Precision-Recall curve

Figure 8 shows two for training loss (train/box_loss and train/cls_loss) and two for validation loss (val/box_loss and val/cls_loss), respectively. These graphs plot the loss over the number of epochs (each epoch representing one full cycle through the training dataset). Box Loss (Figures 8 (a) and (b)): These graphs represent the model's performance in predicting the correct bounding boxes around the objects (localization loss). The training box loss shows a smooth downward trend, indicating that the model is progressively getting better at locating objects during training. The validation box loss fluctuates more, which can be typical if the validation data presents more varied examples or more challenging cases than the training data [17]. Classification Loss (Figures 8 (c) and (d)): These graphs depict the loss associated with the classification accuracy of the objects within the bounding boxes. A steady decline in the training and validation classification loss indicates that the model is improving its ability to classify objects correctly. [18] The solid blue lines represent actual loss values measured at each epoch, while the dotted orange lines show a smoothed trend line, helping to visualize the overall trend amidst the variability from epoch to epoch. Overall, the trained model has highly accurate performance because both training and validation loss decrease over time, converging to a low value.

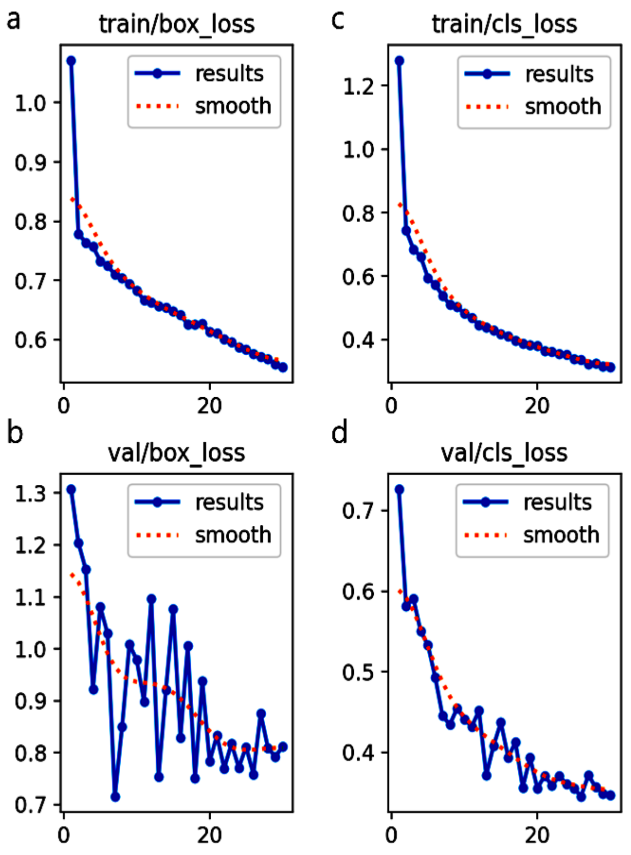


Fig. 8 Training and validation loss: box loss according to (a)Training set and (b)Validation set, class loss according to (c)Training set and (d)Validation set

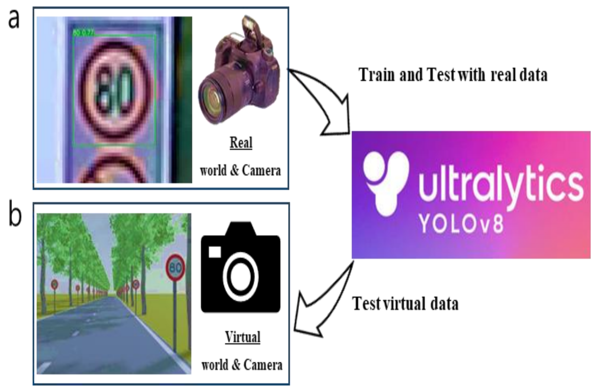


Fig. 9 Virtual data test procedure (a) Real-world & camera (b) Virtual world & camera

3. Virtual Environments Test

Previously, the performance of the YOLO algorithm for speed sign recognition through several indicators has been assessed. The following section will delineate the validation of image data generated within a virtual environment, utilizing an algorithm that the real data have trained. The feasibility of the virtual environment data replacing real-world data will be treated according to the validation results. Figure 9 (a) shows a speed limit sign in the real world, representing a frame from the real dataset taken by a real camera. On the contrary, Figure 9 (b) describes a virtual environment rendering of a road with speed signs to mimic the real one taken by a virtual camera.

3.1. Virtual Test Data Creation

Simcenter Prescan was used to create the virtual environment for the object detection testing. Simcenter Prescan is a platform known for facilitating the development and validation of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles in a virtual environment. [19] The software enables the replication of real-world traffic scenarios, sensor modeling, and the orchestration of various parameters to create complex and dynamic scenes for testing purposes. The virtual camera's position, height, orientation, and Field of View (FoV) can also be adjusted to replicate different perspectives and distances from the objects of interest, like speed limit signs.

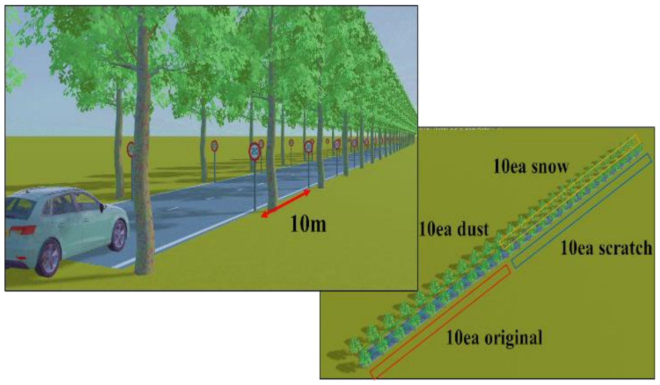


Fig. 10 Virtual data test creation

Table 3. Virtual environment features

Feature		Value
Weather condition		Normal
Camera Specification	FoV	46deg Azimuth 35deg Elevation
	Far clipping distance	300m
	Resolution	1280 × 960 pixel

A virtual camera was installed in a vehicle to take images, as shown in Figure 10. The weather conditions and the specifications of the virtual camera are included in Table 3. The vehicle drives along the straight road with constant speed (36km/h) for 20 seconds. Because the frame rate of the camera is 20Hz, a total of 400 images can be collected. Each speed limit sign is spaced at consistent intervals (as noted by the "10m" marker), allowing the detection system to be assessed on its ability to identify and differentiate between multiple targets at known distances. Since the vehicle is installed, the camera is driving the diverse images of different sizes of signs in a frame, as shown in Figure 11. In addition, reproducing effects such as dust, scratch, and snow are needed to acquire a diverse set of virtual test data. The following effects have means respectively: Dust Effect: Replicating dust involves creating semi-transparent overlays that mimic the appearance of dust on the camera lens or the signs themselves. This can be achieved by adjusting texture properties or by applying particle systems in the virtual environment. Scratch Effect: Scratches can be simulated by adding linear distortions or irregularities to the sign surfaces, which could represent physical wear or damage to the signs or lens imperfections. These can be rendered directly onto the textures of the objects or lens effects on the camera. Snow Effect: To simulate snow, white, semi-opaque particles or layers can be added to the scene. These would not only cover parts of the signs but also float in the air, reducing visibility to replicate the effect of falling or accumulated snow.

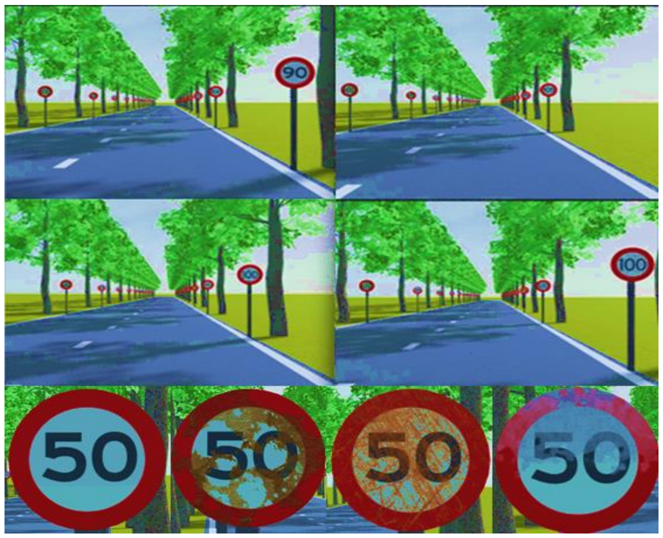


Fig. 11 Samples of the virtual data

3.2. Virtual Test Data Prediction

The trained model, based on the real data, predicts the speed limit signs in the virtual data. Consequently, verifying the algorithm's performance necessitates a manual inspection of the output images. Table 4 summarizes the results of the prediction using virtual test data. There are approximately 70 objects per class in a dataset of 400 images, and the trained YOLO algorithm only correctly identifies ('Detection' includes classification) about 10 of those in each class. A significant number of signs are missing ('No detection'), and some objects are incorrectly identified ('Wrong detection'). As a result, deformation effects such as dust, scratches, and snow did not appear to have any influence. As a consequence, it is easy to overlook that it may be a problem with the detection algorithm itself or with the training method that caused this result. On the other hand, APs of detection cases are around 0.8, and mAP is 0.837, even if the number of well-detected cases is very small. It is necessary to focus on having a high mAP value. According to the image data yielding high AP scores, a notable commonality was discerned. The objects within these images occupied a relatively larger proportion of the image area. This trend aligns with the characteristics observed in the real data used for the training set, where objects of a larger scale were also prevalent.

3.3. Virtual Test Data Modification

While smaller objects are present in the virtual test data, their reduced size might be contributing to the lower detection rate, yet without significantly impacting the precision metric when detections do occur. The initial test data can be described as 'unformatted data', which pertains to virtual images that have not been subjected to any specific formatting or consideration of the training data's image specifications. When the proportion of object size in an image is defined as (a×b)/(x×y)×100%, like in Figure 12, the average of the proportion of training(real) data is 21%, and the smallest one is 3%. On the contrary, the average test(virtual) data is 1.3%, and the smallest one is 0.6%. Namely, the objects in virtual data are ridiculously small compared to real data.

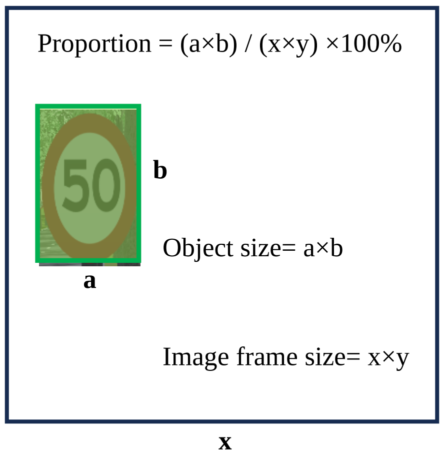


Fig. 12 Proportion of object size to total image frame size

Table 4. Prediction results of virtual test data

Class #	Number of objects	Detection	No detection	Wrong detection	Avg. Precision of detection case
0	68	8	46	14	0.83
1	66	8	48	10	0.85
2	66	7	43	16	0.79
3	72	9	47	16	0.82
4	74	8	54	12	0.89
5	66	10	46	10	0.92
6	74	9	53	12	0.72
7	73	12	47	14	0.82
8	70	9	50	11	0.89
9	68	10	49	9	0.84
Total	697	90	483	124	**mAP:** 0.837

Figures 13 and 14 show the detection results according to the proportion of object size to total image frame size. In practice, once the proportion is smaller than 1%, the objects are missing. When the proportion is equal to 1%, an object is incorrectly identified, even if localization is successful. In contrast, if the proportion reaches 2%, the localization and classification are successful, but the precision is very low, 0.54. When the proportion is larger than 2.8% (Green star in Figure 11), not only does the localization and classification of objects become remarkably accurate, but the precision metric also escalates to exceed a value of 0.9. In short, the data suggest that there is a proportional size threshold below which the detection reliability deteriorates rapidly. Therefore, the unformatted data needs to be modified to 'formatted data' with consideration of the training data's image format, including the size and aspect ratios of objects.

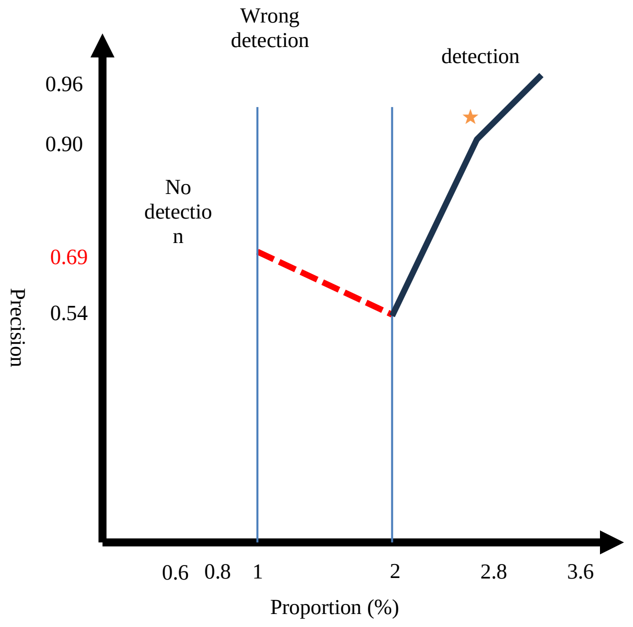


Fig. 13 Precision results according to the proportion

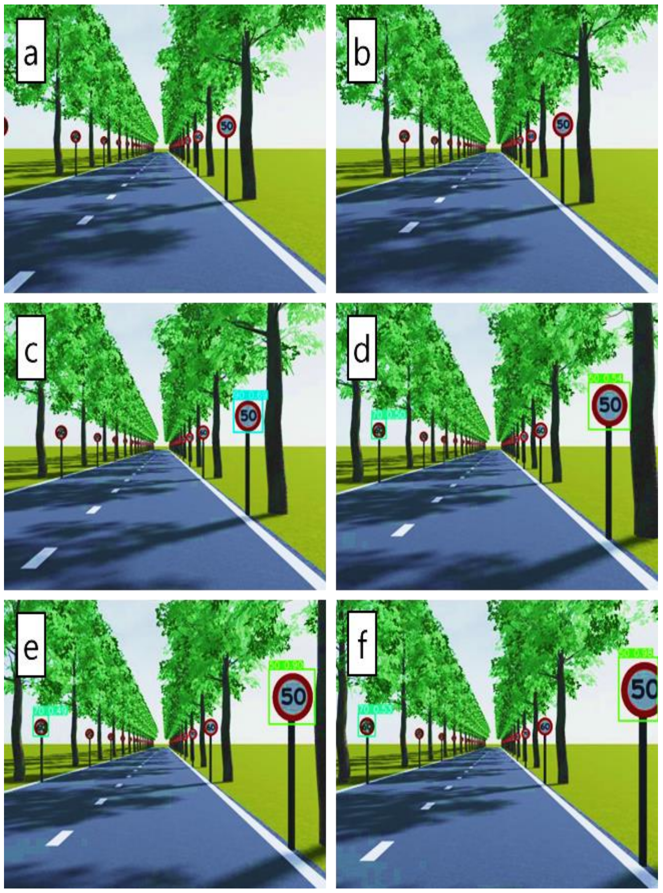


Fig. 14 Virtual images with the object size proportion of (a)0.6%, (b)0.8%, (c)1%, (d)2%, (e)2.8%, (f)3.6%

3.4. Formatted Virtual Test Data Prediction

After collecting 400 images (unformatted images were also 400) with an average proportion of 25% and the smallest one of 2%, the prediction was executed once again. Table 5 presents the results of the predictions.

Table 5. Prediction results of formatted virtual test data

Class #	**Number of objects**	**Detection**	**No detection**	**Wrong detection**	**Avg. Precision of detection case**
1	25	25	0	0	0.94
2	22	22	0	0	0.81
3	23	23	0	0	0.82
4	24	24	0	0	0.82
5	26	26	0	0	0.91
6	22	22	0	0	0.93
7	27	27	0	0	0.82
8	23	23	0	0	0.85
9	22	22	0	0	0.92
10	24	24	0	0	0.91
Total	238	238	0	0	**mAP:** 0.873

With the formatted data, the detection rate improved from previous tests due to increased object sizes, which should have made it easier for the YOLO algorithm to detect and classify the objects correctly. The precision and accuracy of the prediction are also increased compared to the unformatted test data. The results provide a clear contrast in how object size proportion affects the algorithm's ability to accurately detect and classify objects. The average proportion is 25%, higher in the previously used unformatted dataset. The results are likely to show a notable improvement in detection metrics.

4. Conclusion

This paper explored the validation of a speed limit sign recognition system using virtual environments, highlighting the effectiveness of the YOLO algorithm in detecting and classifying speed limit signs under simulated conditions. To bridge the real data and virtual data, the iterative process of testing and refining the dataset for the system model was executed. Moreover, a method of comparing results according to the size and condition of the objects was carried out to analyze statically the model's performance. The study revealed a crucial correlation between the size of objects within image frames and the algorithm's detection performance. In datasets where object proportions were larger, the YOLO algorithm not only achieved higher detection rates but also demonstrated impressive precision. It means that a formatted dataset was constructed by adjusting the proportion of the object's size relative to the original image. However, this research underscores the necessity of conducting a thorough analysis of the training datasets used in each study before generating a virtual test set. The virtual test set closely needs to mimic the characteristics of the training dataset, including aspects such as the position of objects within the images, variations in lighting, and the presence of deformations. This approach ensures that the virtual environment accurately replicates the conditions under which the algorithm was trained, thereby providing a more reliable and relevant evaluation of its performance.

4.1. Discussion and Future Work

Despite its insights, the study encounters several limitations: Limited Diversity in Object Sizes: The initial virtual test data predominantly featured smaller object sizes, which adversely affected the detection performance. Although adjustments were made, the initial results may have skewed perceptions of the algorithm's overall efficiency. Dependence on Virtual Data: The reliance on simulated data, while beneficial for controlling experimental variables, may not fully encapsulate the complexities encountered in real-world scenarios. Real-time YOLO Evaluation: The results from these predictions were not obtained from real-time object detection using a camera in a virtual environment. Therefore, they do not facilitate real-time validation. These limitations suggest that while the findings are indicative of the algorithm's potential performance under controlled conditions, they may not fully capture its efficacy in a live operational setting where dynamic changes and real-time processing are crucial. To overcome these limitations, additional studies will be conducted. More advanced and diverse techniques could be developed for generating virtual data that more closely mimics real-world variations in object sizes and environmental conditions. Integrating more extensive real-world testing could validate the findings from virtual data and help refine the simulation models. Finally, future studies should aim to incorporate real-time testing to more accurately assess the practical applicability of the detection system in real-world scenarios.

By addressing these points, subsequent research can enhance the robustness of detection systems and further solidify the role of virtual testing environments in the development and validation of autonomous driving technologies.

Acknowledgment

This study was supported by the Research Program funded by Seoultech (Seoul National University of Science and Technology).

References

[1] Jakkree Srinonchat, "Efficient Detection of Speed Limit Signs within Obscure Environment," *IET 3rd International Conference on Wireless, Mobile and Multimedia Networks (ICWMNN 2010)*, Beijing, pp. 311-314, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[2] Yogesh Valeja et al., "Traffic Sign Detection Using Clara and Yolo in Python," *2021 7th International Conference on Advanced Computing and Communication Systems*, Coimbatore, India, pp. 367-371, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Christian Birchler et al., "Cost-Effective Simulation-Based Test Selection in Self-Driving Cars Software with SDC-Scissor," *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Honolulu, HI, USA, pp. 164-168, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Yue Kang, Hang Yin, and Christian Berger, "Test Your Self-Driving Algorithm: An Overview of Publicly Available Driving Datasets and Virtual Testing Environments," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 171-185, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] Joseph Redmon, and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 6517-6525, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] Tafreed Ahmed et al., "The YOLOv8 Edge: Harnessing Custom Datasets for Superior Real-Time Detection," *2023 18th International Conference on Emerging Technologies*, Peshawar, Pakistan, pp. 38-43, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Saurabh Pujar et al., "Invited: Automated Code Generation for Information Technology Tasks in YAML through Large Language Models," *2023 60th ACM/IEEE Design Automation Conference*, San Francisco, CA, USA, pp. 1-4, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Everything you need to build and deploy computer vision models, Roboflow. [Online]. Available: https://roboflow.com

[9] Yang, Speed Limit Detection Computer Vision Project, Roboflow. [Online]. Available: https://universe.roboflow.com/yang-ygcec/speed-limit-detection

[10] Kanjar De, and Marius Pedersen, "Impact of Colour on Robustness of Deep Neural Networks," *2021 IEEE/CVF International Conference on Computer Vision Workshops*, Montreal, BC, Canada, pp. 21-30, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Performance Metrics Deep Dive, Ultralytics. [Online]. Available: https://docs.ultralytics.com/guides/yolo-performance-metrics/

[12] Tsung-Yi Lin et al., "Microsoft Coco: Common Objects in Context," *Computer Vision–ECCV 2014: 13th European Conference*, Zurich, Switzerland, pp. 740-755, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ekaba Bisong, *Google Colaboratory*, Building Machine Learning and Deep Learning Models on Google Cloud Platform, Apress, Berkeley, CA, pp. 59-64, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] Jesse Jon Davis, and Mark Harlan Goadrich, "The Relationship between Precision-Recall and ROC Curves," *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh Pennsylvania USA, pp. 233-240, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[15] Stéphan Clémençon, and Nicolas Vayatis, "Nonparametric Estimation of the Precision-Recall Curve," *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal Quebec Canada, pp. 185-192, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[16] Kendrick Boyd, Kevin H. Eng, and C. David Page, "Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals," *Machine Learning and Knowledge Discovery in Databases: European Conference*, Prague, pp. 451-466, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[17] Jiahui Yu et al., "UnitBox: An Advanced Object Detection Network," *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam The Netherlands, pp. 516-520, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[18] Yue Wu et al., "Rethinking Classification and Localization for Object Detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp. 10186-10195, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] Siemens Digital Industries Software, Siemens. [Online]. Available: https://www.sw.siemens.com/en-US/