Bunch-of-Keys Module for Optimizing a Single Image Detector Based on the Property of Sequential Images

Wonwoo Lee¹, Yoonsuk Choi², Jeesu Kim³, Member IEEE, Jinwoo Yoo⁴, Member IEEE

¹Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Republic of Korea
²Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Republic of Korea
³Department of Cognomechatronics Engineering, Pusan National University, Busan 46241, Republic of Korea
⁴Department of Automotive Engineering, Kookmin University, Seoul, 02707, Republic of Korea

Corresponding author: Jiwon Yoo (e-mail: jwyoo@kookmin.ac.kr).

This work was supported by two National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) [NRF-2021R1F1A1062153 and NRF-2021R1A5A1032937].

ABSTRACT In image processing, deep learning networks have been continuously developed and are used in many fields. However, most networks do not reflect image continuity. In this paper, we propose a novel bunch-of-keys module connected to the backend of a deep learning network to improve the detector performance on sequential images. This module optimizes existing deep learning networks to detect sequential images without retraining. This procedure reduces time and computing costs, and the average precision improves with a minimal drop in the frames per second. By adopting a sliding window method that uses three consecutive images, the keys are generated by comparing the positions of the detected boxes for each of the images using generalized intersection over union. The two key types perform correcting operations. The rectifying key has the effect of adding or merging undetected bounding boxes in mid-frame. The tracking key has the effect of compensating for bounding boxes lost for no reason in the third frame. The candidate box extracted using each key determines whether to add or merge to the target image in the correction task. This task calculates the complete intersection over union (CIoU) score between candidate boxes and all boxes of the target image and is divided into add or merge cases according to a set of CIoU criterion. As a result of adding or merging the bounding box to the missing object, detection performance was improved up to 3% in terms of the average precision.

INDEX TERMS Multiple object tracking, real-time tracking, matching keys, tracking by using keys, improving deep neural network performance, post-processing module.

I. INTRODUCTION

Detecting objects in images and classifying them has evolved from a method using traditional techniques to a method using deep learning. By introducing convolution, classifying images (while preserving the structural meaning of the images), detecting the object position within the image, and classifying the class has become possible.

Before AlexNet [1] was proposed, it had been difficult to use deep learning network because of considerable computation costs. However, after AlexNet, deep learning made rapid progress, and research has been conducted on various topics, such as network structure, image preprocessing, and data augmentation. For example, Shin et al. improved the performance of a deep learning network by generating training data using a generative adversarial network to better detect polyp in colonoscopy [22]. In addition, Kim et al. proposed the Local Augment method [23], which highly alters the local bias property so could generate significantly diverse augmented images and offer the network with a better augmentation effect.

In addition, research using deep learning is actively conducted regardless of the field, such as medical care, transportation, and aviation. In particular, in the field of autonomous driving, since real-time detectors, such as You Only Look Once (YOLO) [9], Tinier-YOLO [24] and Single Shot Multibox Detector (SSD) [10], were proposed, various studies to use the camera have been conducted.

As most real-time networks process continuous images, the characteristics of continuous images must be used. The researched tracking algorithm is specialized for continuous
images. For example, research is underway to assign IDs and track objects using detectors [15] trained on sequential images. Similarly, a study on multiple object tracking considering the shape, structure, motion, and size for more accurate tracking is in progress [20]. Additionally, to reduce the computational amount of tracking, a study is being conducted to significantly reduce the amount of computation required for tracking by estimating the size and angle of the target bounding box using only a single search region [21].

However, the tracking algorithm traces and detects objects even if obscured by other objects. Thus, we cannot use the tracking algorithm to detect only visible objects in the image. As most networks are trained with image datasets of individual situations, they cannot detect objects in a continuous situation. When using image datasets for sequential images, a dataset must be created by labeling and a network must be retraining incurring a high cost.

In this paper, we propose a bunch-of-keys (BOK) network that can improve performance by combining information extracted from consecutive images without retraining the trained network. The BOK method introduces a key that compares and matches objects detected in successive images and works by attaching it to the back of the existing network as a module. Thus, there is absolutely no cost for additional labeling or training. In addition, the computational cost is meager, so the decrease in frames per second (FPS) is also small.

The contributions of this paper to the field of deep learning are as follows:

- Improving the performance of the network in successive images,
- Providing a module that can be used regardless of network components, and
- Needlesness of retraining the existing network.

The paper is organized as follows. Related work about backbone network, real-time detectors and bounding box comparison algorithm are introduced in Section II. In Section III, BOK module is proposed including the motivation and the architecture of network, key generation algorithm and correction task. In Section IV, the first experimental results are given, where the variation of average precision (AP) and FPS according to the box comparison algorithm used with our module are compared. The second results verify the performance when various existing networks are used. Furthermore, algorithms for optimizing the hyperparameters used in the module are also provided in Section IV. Finally, the conclusion is drawn in Section V.

II. Related Work
A trained detector and a comparison method for the detected boxes are needed to generate a key in the sequential images. As many studies exist on comparative methods, we employed these studies in conducting this research.

A. Deep Learning Network

### 1) NETWORKS USED AS BACKBONES
Alex et al. proposed AlexNet [1], exhibiting overwhelming performance with a top-five error rate of 15.4% in the classification task, where no significant performance improvement previously existed. AlexNet can achieve good performance using a deep convolutional neural network (CNN) for image classification.

Karen et al. proposed VGGNet [2] with a deeper structure than the existing AlexNet using a CNN and ReLU activation function. This network has a simple structure, is easy to learn, and exhibits excellent performance, so it has been widely used as a backbone for other networks. However, it consumes substantial memory due to the number of parameters.

Szegedy et al. designed the inception module by referring to the Network In Network [3]. They proposed GoogleNet [4] to combine several small convolutional layers into one module and merge the results.

He et al. [5] suggested ResNet, which introduces the skip connection because a deeper, wider deep learning network structure has better performance but is more challenging to train due to the many parameters. In addition, ResNet achieved a top-five error rate of 3.57% in the ImageNet task using an ensemble configuration stacked up to 152 layers, more than eight times deeper than the existing VGG16 network.

Tan et al. proposed a compound coefficient, a new scaling method for dimensions of depth, width, and resolution. Furthermore, a baseline network was designed using neural architecture search, and the EfficientNet [25] was designed by scaling up this baseline network. In particular, EfficientNet-B7 performed 6.1 times faster than the latest convolution networks on the ImageNet dataset.

### 2) REAL-TIME DETECTOR
The Fast Region-based Convolutional Network method (Fast R-CNN) [7] improved in speed by performing classification and bounding box regression simultaneously, instead of using the R-CNN method that separately performs classification and bounding box regression after the region proposal. The Faster R-CNN [8] is a real-time network that introduces the Region Proposal Network (RPN) to solve the Fast R-CNN problem of taking a long time when using the selective search. The RPN method obtains region proposals by training the convolutional layer. This network is accurate enough to detect overlapping objects, but is slower than other real-time detectors.

Redmond et al. proposed YOLO [9], a representative algorithm for one-stage networks. In addition, YOLO is very fast by segmenting the input image into a grid and performing

| Table I | Real-time detector comparison |
|---|---|
| Faster RCNN [8] | YOLO [9] | YOLOv4 [19] | SSD [10] |
| **Real-Time detector** | ✓ | ✓ | ✓ | ✓ |
| **Multi object detector** | ✓ | ✓ | ✓ | ✓ |
| **1-Stage detector** | ✓ | ✓ | ✓ | ✓ |
| **Using grids method** | ✓ | ✓ | ✓ | ✓ |
| **Detecting objects in multi feature maps** | ✓ | ✓ | ✓ | ✓ |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
bounding box regression and classification for each cell. YOLOv2 [17] increased the size of the grid map and concatenates the high-resolution feature map and the low-resolution feature map to compensate for the insufficient performance of YOLO. This work was to better predict the various sizes. YOLOv3 [18] used independent logistic classifiers instead of softmax for class prediction considering multi-label. It used three bounding boxes and three feature maps to better detect objects of various scales. YOLOv4 [19] was a network that applied various latest technologies to YOLO. Bochkovskiy et al. introduced Bag of Freebies in training, and improved network detection performance using Bag of specials. This state-of-the-art network has excellent speed and accuracy. This network is easy to use, fast, and relatively accurate, but it does not detect overlapping objects very well.

Liu et al. proposed SSD [10], a one-stage network with fast speed. As the convolutional operation progresses, the feature map becomes smaller. Thus, the SSD uses same-size anchor boxes to detect small objects in shallow layers and large objects in deep layers in the CNN. Although this network is relatively fast and easy to control performance and speed, it has the disadvantage of being relatively difficult to use. Table I is attached to summarize the characteristics of real-time detectors.

B. Bounding Box Comparison Algorithm

1) LOSS FUNCTION
Most deep learning networks primarily use the L1 and L2 norms as loss functions for bounding box regression. Bounding box regression uses x and y (the center box coordinates) as the parameters and width and height, denoting the ratio of the size of the box input image. The convergence result is evaluated using the intersection over union (IoU) between the x, y, width, and height of the predicted box and the $x^{gt}$, $y^{gt}$, width$^{gt}$, and height$^{gt}$ of the ground truth.

Rezatofighi et al. proposed the generalized IoU (GIoU) [11] and introduced it as a loss function. The IoU cannot be used for learning because the values are zero when the boxes are not overlapped. In the GIoU, a score can be calculated even if the boxes do not overlap by introducing Area C of the area surrounding the predicted bounding box and bounding box$^{gt}$.

The complete IoU (CIoU), proposed by Zheng et al. [12], increased the convergence speed by introducing the distance between the center points of two boxes and the ratio of the width and height between the two boxes as parameters instead of C to compensate for the slow convergence speed of the GIoU.

2) IMAGE COMPARISON ALGORITHM
The mean squared error is the simplest method to calculate the similarity between images. However, it is difficult to reflect the pixel structure or relative value.

Figure 1. Schematic diagram of the Bunch of Keys network. It improves network performance by detecting continuous images from a monocular camera in the existing network. Results detected from the continuous images are stored sequentially in bounding boxes: A, B, and C. The continuity of the location information generates a key in which information about the same object is stored. A candidate box is created to be drawn on the target image using the key. If no box in the target image is similar to the candidate box, add is performed; otherwise, merge is performed. The intermediate image (Image B) is output. The detection result of the next image is input using a sliding window method. This network iterates until no next input image exists.
The existing deep learning network is designed to process individual images unrelated to each other, not continuous images. Most deep learning networks are trained to improve performance by connecting the BOK module behind the existing network. The existing network can be used to adapt to continuous images by improving performance without additional training.

This study assumes that an object cannot disappear from continuous images of 10 FPS or more, except in special cases, such as slowly moving out of the image or moving away from the camera. Therefore, the performance of the network was improved by detecting and adding missing bounding boxes. The BOK network (Fig. 1) is designed to group and process three consecutive images in the sliding window method. A key is generated by matching detected bounding boxes, and a missing bounding box is added using this key. Because the computing cost of this method is low, network performance can be improved with very little FPS loss. Proof of this is presented in the experimental results in Section IV.

A. BUNCH OF KEYS NETWORK ARCHITECTURE

The existing deep learning network method is used to process the video in the real-time network when the video comes in at a speed of 10 FPS or higher from the monocular camera. However, the fatal disadvantage of real-time networks is that a higher speed leads to lower network performance. In this paper, network performance is improved using the continuity of the results (bounding boxes) detected by the network. The BOK, suitable for continuous image processing, employs the relationship between information on the detected bounding boxes, which is the network output. Because it does not require computation involving many variables and does not use special formulas, it can be used as a network module to improve performance with a slight drop in FPS.

Fig. 1 reveals that the detection result of the first image is output without correction when two images enter an existing network for the first time. From the time the detection results of the three images enter the module, the BOK compares the detection results of the three images using a sliding window. The information is stored in the key vector if the same object exists due to comparing the detected bounding boxes. Various key vectors can be created according to their purpose.

In this paper, we created and used a rectifying key to correct Image B, the second image in the sliding window, and a

### Key generation code

#### Algorithm I

**Key generation pseudocode**

```
# Only elements greater than 0 and less than the threshold are converted to 1.
Glu = (Glu_key > 0) and (Glu_key < Glu_THRESHOLD)

# In the Glu matrix, an element with a value of 1 is replaced with a number corresponding to each column number. The row of the Glu matrix matches Bboxes 1 and the column matches Bboxes 2.
FOR i is 1 to the number of bounding boxes 1
FOR j is 1 to the number of bounding boxes 2
    IF Glu(i, j) == 1 THEN
        key_save(i, j) = j;
    ELSE
        key_save(i, j) = 0;
    ENDIF
ENDFOR
ENDFOR

# key_save matrix is compressed into a one-column vector. The column number with the index of Element 1 is stored in the created vector. The index number of the vector is the same as the number of Bboxes 1, and the element data are the same as the number of Bboxes 2.
KEY1 = column_max(key_save)
```
B. KEY GENERATION ALGORITHM

Each successive image is detected using a deep learning network. The set of detected bounding boxes for each image is called Bboxes, where \( i \) boxes are created in Bboxes \( A \), \( j \) boxes are created in Bboxes \( B \), and \( k \) boxes are created in Bboxes \( C \) (\( i, j, \) and \( k \) are the number of boxes). In Fig. 2(a), each arrow represents a comparison of the detected bounding boxes. The rectifying key (Fig. 2(b)) is created through the relationship between the red arrow (Bboxes \( A \to Bboxes \; B \)) and blue arrow (Bboxes \( A \to Bboxes \; C \)). Similarly, the tracking key (Fig. 2(c)) is created by considering the relationship between the red and green arrows (Bboxes \( B \to Bboxes \; C \)). When comparing the bounding boxes between each image, the score was calculated using the GIoU [11] equation, as follows:

\[
\text{GIoU} = IoU - \frac{Z}{|Z|} \frac{|Z|}{|X \cup Y|} \tag{1}
\]

where \( Z \) is the area of the smallest box enclosing Boxes \( X \) and \( Y \), and \( Z / (X \cup Y) \) indicates the area remaining in Area \( Z \), excluding the area occupied by Boxes \( A \) and \( B \). The calculated GIoU score has a range of \(-1 < \text{GIoU} \leq 1\).

For the rectifying key, the size of the matrix representing the relationship for each bounding box becomes an \((i \times j)\) matrix and \((i \times k)\) matrix, and for the tracking key, it becomes a \((j \times i)\) and \((j \times k)\) matrix. After leaving only the maximum values of each row and column in each matrix, the key can be generated by filtering with the GIoU limit, a preset threshold, and compressing this matrix based on the column. The detailed MATLAB code for generating the key is described in Algorithm I.

Since the score is compared to seventh decimal places, the probability of getting the same score is very low. However, if more than one element with the highest score is detected, the box with the closest midpoint distance is matched. If the distances between the midpoints are also the same, a random box is matched.
In Fig. 2(a), the first row (red area) of generated keys is rectifying Key 1 generated by comparing Images A and B, and the second row (blue area) is rectifying Key 2 generated by comparing Images A and C. In addition, in Fig. 2(b), the first row (red area) of tracking keys is tracking Key 1 generated through the comparison of Images B and A. The second row (green area) is tracking Key 2 generated through a comparison of Images B and C. The rectifying Key 1 and tracking Key 1 are matrices created from the same matrix (comparison of Images A and B) but compressed in different directions depending on their purpose.

1) RECTIFYING KEY
The key generation and usage are described in Fig. 3. Each image, A, B, and C is a series of images processed in a sliding window with a short time difference. In other words, after processing Images 1 to 3, Images 2 to 4 are processed. The boxes drawn in each image are bounding boxes detected by the detector. Each of these bounding boxes includes the coordinates and size information of the box in the image, such as x, y, width, and height. In the process of Fig. 3(a), this information is used to calculate the bounding box matching score. Then, a key is generated based on the compared information.

In the generated rectifying key, each column number is the same as the bounding box number detected in Image A. If there are four boxes detected in Image A, the number of rectifying key columns is four.

Additionally, in the rectifying key, the box number of Bboxes B matched to the box of Bboxes A of the corresponding column number is stored in Row 1, and the box number of Bboxes C is stored in Row 2. For example, from Column 4 of the table in Fig. 3(d), Box 4 of Bboxes A, Box 3 of Bboxes B, and Box 3 of Bboxes C are matched. Moreover, Column 2 indicates that Box 2 of Bboxes A is not matched with any Box of Bboxes B but is matched to Box 1 of Bboxes C.

Based on the information inside the rectifying key, cases can be extracted where the same bounding box was detected in Images A and C but not B. In other words, a case in which
the object detected in Images A and C is not detected in B in the continuous image. Because it was assumed that a situation in which the bounding box is lost in the middle could not occur in continuous images of 10 FPS or more, a correction operation (bounding box adding or merging) is performed on Image B (Fig. 3(e)).

2) TRACKING KEY

The creation process of the tracking key is like that of the rectifying key. As in Fig. 3(a), boxes are compared using the sliding window method. In Fig. 3(b), the difference is that the size of the generated column is the same as the number of bounding boxes in Image B. The box numbers in each row are for Images A and C, and the image to be corrected is Image C. In the tracking key, it is possible to extract cases where the matched boxes from Images A and B do not exist in Image C. These cases include one of the following three situations:

- when an object leaves the image and disappears,
- when an object obscures another object, and
- when the object is not correctly detected in the third image.

In the first and second situations, false detection occurs if a bounding box is drawn on the third image. As a solution to the first situation, an area factor of 0.8 was introduced so that the tracking key can operate only in the central 80% of the image area. The following correction task can solve the second situation. In the third case, a problem occurred in which the bounding box was not detected. Therefore, network performance can be improved by drawing the box on Image C (Fig. 3(c)).

3) PREVENTION OF ERROR ACCUMULATION WHEN USING THE TRACKING KEY

When using the tracking key, linear interpolation was used to reduce the amount of computation. This method has a high probability of causing an error when the camera viewpoint changes rapidly or when an object rapidly moving horizontally is detected. A candidate box drawn using the proposed tracking key must be matched with the box in Image A in the next cycle, and a new box is drawn in Image C. When an erroneously detected object occurs, errors continue to accumulate even afterward. This error can be resolved using an image comparison algorithm, but a severe drop in FPS occurs.

For example, in the first loop of Fig. 4(a), the tracking key adds two candidate boxes (Boxes A and B) to Image 3. In the second loop, the added box is matched with Boxes A and B in Image 2, generating a key. However, if the object position changes rapidly, as in Image 4, the tracking key determines that it is a different object and redraws the box, which is erroneous, so the accuracy reduces.

The proposed BOK algorithm did not consider the box created in the previous loop in the tracking key generation step to avoid this. In Fig. 4(b), Boxes A and B added in the first loop were not considered in the second loop, and only Box C was matched to the tracking key and drawn in Image 4. This method avoids error accumulation problems when using tracking keys with a small amount of computation.

4) KEY REMAPPING OPERATION WITH RGB

As the key is created with only location information, this method cannot determine when another object of the same class is detected in the same location. However, color information also exists in the image; thus, a more precise key
Detection results according to matching algorithm (one class - car)

| Network        | Method                        | Average precision | Improvement | Frames per second |
|----------------|-------------------------------|-------------------|-------------|------------------|
| ResNet50 & YOLOv2 [17] | -                             | 57.99             | -           | 41.1             |
|                 | Dist & FSImc [14]             | 58.00             | 0.02%       | 24.0             |
|                 | Dist & CIoU [12]              | 58.17             | 0.31%       | 39.3             |
|                 | IoU & FSImc [14]              | 58.00             | 0.02%       | 10.6             |
|                 | GIoU [11] & CIoU [12]         | **58.39**         | **0.69%**   | **38.4**         |
| ResNet101 & YOLOv2 [17]  | -                             | 72.25             | -           | 31.7             |
|                 | Dist & FSImc [14]             | 72.49             | 0.33%       | 20.2             |
|                 | Dist & CIoU [12]              | 72.62             | 0.51%       | 30.8             |
|                 | IoU & FSImc [14]              | 72.39             | 0.19%       | 7.22             |
|                 | GIoU [11] & CIoU [12]         | **72.63**         | **0.53%**   | **30.3**         |
| ResNet50 & SSD [10]        | -                             | 41.61             | -           | 35.2             |
|                 | Dist & FSImc [14]             | 41.74             | 0.31%       | 23.1             |
|                 | Dist & CIoU [12]              | 41.73             | 0.29%       | 29.3             |
|                 | IoU & FSImc [14]              | 41.70             | 0.22%       | 6.84             |
|                 | GIoU [11] & CIoU [12]         | **41.88**         | **0.65%**   | **33.43**        |
| ResNet101 & SSD [10]        | -                             | 50.32             | -           | 28.6             |
|                 | Dist & FSImc [14]             | 50.61             | 0.58%       | 22.4             |
|                 | Dist & CIoU [12]              | 50.53             | 0.42%       | 28.3             |
|                 | IoU & FSImc [14]              | 50.47             | 0.30%       | 10.3             |
|                 | GIoU [11] & CIoU [12]         | **50.81**         | **0.97%**   | **27.9**         |

C. CORRECTION TASK

A candidate box drawn on each target image can be created using the two previously generated keys (rectifying and tracking keys). The candidate boxes are made from the average value of the parameters (x, y, width, and height) of each matched box, which prevents the problem that the FPS decreases as the computation increases. Before adding candidate boxes to the target image, we must consider the possibility that the same object was detected but judged to be a different object due to the difference in the size or proportion of the detected boxes. In this case, adding a box causes over-detection, decreasing the AP. To prevent this, the CIoU [12], a filter that can classify key-based candidate boxes as cases of adding and merging, is introduced to determine whether to add a box. The CIoU uses a method that reflects the ratio of boxes as a parameter and employs the GIoU, which introduces the distance between boxes. Therefore, it becomes a criterion for defining the relationship between boxes more sensitively than the GIoU. The definition of the CIoU [12] can be calculated as follows:

\[
CIoU = 1 - IoU + \frac{\rho(b,b^*)}{c^2} + \alpha \nu, \tag{2}
\]

where \(\alpha\) is a trade-off parameter, and \(\nu\) is a parameter for indicating the continuity of the box-width ratio. Using (2), we calculate the CIoU scores of candidate boxes and all bounding boxes in the target image and check whether a higher value exists than the CIoU criterion, a preset threshold. If a box with a CIoU criterion or higher score exists in the target image, it is judged as the same object, and no box is added. The unadded boxes are merged with the box with the highest CIoU score in the target image set to further improve the AP.

1) DRAWING CANDIDATE BOX OPERATION

In Section III-C, we used the key to determine the missing box in the target image and created a candidate box to add. In addition, using the CIoU, the boxes to be merged without adding were filtered from the extracted boxes. After this process, the remaining unfiltered boxes were added to the target image (green box in Fig. 3(c) and red box in Fig. 3(e)) to improve the AP. For the box parameters to be added, linear interpolation prevents an FPS decline due to increased computation.

2) MERGING OPERATION

In Section III-C, the filtered candidate box with a higher score than the CIoU criterion is merged with the box with the highest matched score in Image B. A candidate box is calculated as the average value of the x, y, width, and height values of the boxes in the images for A, B, and C to replace the existing box in the target image (blue box in Fig. 3(e)).

IV. Experiment

In this paper, two experiments check the optimal bounding box comparison algorithm and determine the improvement in network performance:
Experiment 1: This experiment verifies which combination of algorithms is effective when mapping keys and performing correction tasks. We used YOLOv2 [17] and SSD networks, trained with only one class (car).

Experiment 2: This experiment demonstrates the performance improvement of networks trained using multiple classes (car, human, and truck) with the algorithm combination verified in Experiment 1. The method is introduced to determine the optimal hyperparameter value.

### A. Checking the Performance of the Combination of Comparison Algorithms

#### 1) EXPERIMENTAL SETTING

This study employed MATLAB 2021a and was tested in the central processing unit (CPU) AMD Ryzen 7 3700X and graphics processing unit (GPU) RTX 3090/RAM: 128 GB environment. Each network was trained using the dataset provided by KITTI [16] and COCO, and the training options for each network were set to an initial learning rate of 0.001, a learning rate drop factor of 0.01, a learning rate drop period of 5, a mini-batch size of 16, and 120 maximum epochs. Only the rectifying key was used to shorten the experiment time, and the 0020 dataset of the KITTI tracking datasets was used as the testing dataset. The purpose of this paper is detection, not object tracking; thus, the boundary box that should not be detected in the existing ground truth is deleted (e.g., the part that is not visible because it is covered by a vehicle or the part that detects an object that has already left the image).

#### Table III

Comparison of detection result (three classes: car, human, and truck)

| Dataset | Network | Algorithm       | mAP   | Classes | Improvement | FPS |
|---------|---------|-----------------|-------|---------|-------------|-----|
|         |         |                 |       |         | Car | Human | Truck |       |       |       |
| KITTI   | Multi-object tracking |             |       |         | Ref | 71.97 | 81.69 | 41.93 | 92.28 | -     | 31.48 |
| 0009 [16] | ResNet50 & YOLOv2 [17] | Rectifying | 72.14 | 81.89 | 41.93 | 92.61 | 0.24% | 26.59 |
|         |         | Tracking        | 72.16 | 81.77 | 41.93 | 92.77 | 0.26% | 28.51 |
|         |         | Rectifying & Tracking | **72.21** | 81.93 | 41.93 | 92.77 | **0.34%** | 26.90 |
|         |         |                 |       |         | Ref | 69.43 | 81.40 | 34.00 | 92.90 | -     | 23.45 |
|         |         | Rectifying      | 69.70 | 81.81 | 33.95 | 93.36 | 0.39% | 21.92 |
|         |         | Tracking        | 69.68 | 81.31 | 33.89 | 93.84 | 0.36% | 21.48 |
|         |         | Rectifying & Tracking | **69.96** | 82.14 | 33.89 | 93.84 | **0.76%** | 20.04 |
|         |         |                 |       |         | Ref | 47.70 | 55.98 | 1.14 | 85.96 | -     | 1.414 |
|         |         | Rectifying      | 48.29 | 56.40 | 1.14 | 87.31 | 1.24% | 1.390 |
|         |         | Tracking        | 48.35 | 56.29 | 1.14 | 87.63 | 1.38% | 1.372 |
|         |         | Rectifying & Tracking | **48.49** | 56.33 | 1.14 | 87.99 | **1.67%** | 1.365 |
|         |         |                 |       |         | Ref | 53.83 | 68.66 | 1.72 | 91.10 | -     | 3.311 |
|         |         | Rectifying      | 53.96 | 68.96 | 1.72 | 91.20 | 0.25% | 3.293 |
|         |         | Tracking        | 54.11 | 69.23 | 1.72 | 91.37 | 0.52% | 3.266 |
|         |         | Rectifying & Tracking | **54.16** | 69.44 | 1.72 | 91.33 | **0.63%** | 3.242 |
| KITTI   | Multi-object tracking |             |       |         | Ref | 66.06 | 92.17 | 58.46 | 47.55 | -     | 31.50 |
| 0015 [16] | ResNet50 & YOLOv2 [17] | Rectifying | 66.33 | 92.50 | 58.93 | 47.55 | 0.40% | 27.22 |
|         |         | Tracking        | 67.79 | 92.92 | 59.04 | 51.42 | 2.62% | 27.13 |
|         |         | Rectifying & Tracking | **68.06** | 93.34 | 59.41 | 51.42 | **3.02%** | 27.10 |
|         |         |                 |       |         | Ref | 69.07 | 93.20 | 59.80 | 54.20 | -     | 22.10 |
|         |         | Rectifying      | 69.38 | 93.75 | 60.23 | 54.17 | 0.46% | 20.32 |
|         |         | Tracking        | 70.65 | 93.72 | 60.47 | 57.76 | 2.29% | 21.68 |
|         |         | Rectifying & Tracking | **70.93** | 94.01 | 61.02 | 57.76 | **2.69%** | 20.33 |
|         |         |                 |       |         | Ref | 38.25 | 83.88 | 30.88 | 0    | -     | 1.611 |
|         |         | Rectifying      | 38.53 | 84.27 | 31.32 | 0   | 0.71% | 1.532 |
|         |         | Tracking        | 38.40 | 84.06 | 31.13 | 0   | 0.38% | 1.305 |
|         |         | Rectifying & Tracking | **38.65** | 84.55 | 31.40 | 0   | **1.03%** | 1.414 |
|         |         |                 |       |         | Ref | 39.70 | 90.88 | 28.21 | 0    | -     | 3.362 |
|         |         | Rectifying      | 40.14 | 91.38 | 29.04 | 0   | 1.12% | 3.248 |
|         |         | Tracking        | 40.18 | 91.56 | 28.96 | 0   | 1.21% | 3.196 |
|         |         | Rectifying & Tracking | **40.57** | 91.95 | 29.75 | 0 | **2.19%** | 3.239 |

#### Figure 6

Plot of the average precision (AP) with hyperparameters used in the bunch of keys network. Both generalized intersection of union (IoU) and complete IoU greatly affect the average precision; thus, they exhibit a mountain-like shape.
2) FILTER ALGORITHM SELECTION FOR KEY GENERATION

Table II depicts the results of classifying continuous images using only the rectifying key for the proposed networks. Since the test dataset was constructed in a driving environment, we had to use a real-time detector. Therefore, we used YOLO and SSD, which are representative real-time detection networks. YOLOv2 and SSD provided by MATLAB were used as a reference network, and ResNet50 and ResNet101 were used for the backbone.

The following algorithms were combined and used as filters for the key mapping and add/merge judgment:
- the Euclidean distance between boxes (Dist);
- FSIMc [14], which calculates the similarity for each image, including color information;
- IoU, indicating the degree of overlap between boxes;
- generalized IoU, introducing the distance between boxes in IoU [11];
- complete IoU [12], introducing the aspect ratio of the box as a parameter for the GIoU.

In the experiment, the case applying the GIoU and CIoU filters to ResNet101 and YOLOv2 recorded the highest AP. The FPS loss was only about 4.4%. Thus, the AP can be improved only by post-processing without changing the network configuration with only a slight loss of FPS. The AP increased in all cases, and ResNet-101 & SSD (0.97%) exhibited the highest improvement.

Fig. 5(a) is the result of detection only with the existing network, and 5(b) is the result of applying the BOK network. The yellow boxes are detected using the existing reference network (ResNet101 and YOLOv2), and the red box is added based on the detected result. In Fig. 5(a), the previously detected vehicle is not detected in Image 2, and the proposed network detects the corresponding part and adds a box.

Fig. 6 is a graph plotting the AP distribution according to the threshold of each filter. Both GIoU and CIoU greatly influence the AP and appear mountain-like so that the optimal value can be found through sampling.

| Dataset                  | Network                  | Mean average precision | Number of searched points |
|--------------------------|--------------------------|------------------------|--------------------------|
|                          |                          | Using GIoU = 0.5, CIoU = 0.5 | Using optimized GIoU, CIoU |
| KITTI Multi-object tracking 0009 [16] | ResNet50 & YOLOv2 [17] | 70.44                  | 72.21                    | 260                  |
|                          | ResNet101 & YOLOv2 [17] | 69.53                  | 69.96                    | 88                    |
|                          | DarkNet50 & YOLOv3 [18] | 48.18                  | 48.49                    | 122                   |
|                          | CSPDarkNet53 & YOLOv4 [19]| 53.98                  | 54.16                    | 120                   |
| KITTI Multi-object tracking 0015 [16] | ResNet50 & YOLOv2 [17] | 66.38                  | 68.06                    | 228                   |
|                          | ResNet101 & YOLOv2 [17] | 70.82                  | 70.93                    | 88                    |
|                          | DarkNet50 & YOLOv3 [18] | 38.40                  | 38.65                    | 83                    |
|                          | CSPDarkNet53 & YOLOv4 [19]| 40.28                  | 40.57                    | 154                   |

2. Bunch of Keys Network Performance Verification

1) EXPERIMENTAL SETTING

The KITTI tracking dataset was used for the testing dataset in this experiment, and the boundary box that should not be detected in the existing ground truth is deleted. For training ResNet50 with YOLOv2 and ResNet101 with YOLOv2, the left color images of the KITTI 2D object detection dataset were used. In addition, 80% of the 7,481 sheets were used as training datasets, whereas 20% were used as validation datasets. The network training options are the same as in Section IV-A except for 400 maximum epochs.

Moreover, for verification in various networks ResNet50 with YOLOv3 and DarkNet53 with YOLOv4 were installed and used as add-ons provided by MathWorks. YOLOv4 was used for verification on the latest network. These two networks were trained with COCO; thus, they were adjusted to the three
classes used in the above experiment (car→car; bus, truck, or train→truck; and person→human).

2) PERFORMANCE VERIFICATION OF THE BUNCH OF KEYS NETWORK

Table III lists the results of applying the BOK network proposed in this paper to various networks. The mean AP (mAP) increased in experiment cases. In addition, the drop in FPS is very slight. In most cases, it was possible to increase the mAP with an FPS loss of about 10%. In particular, when BOK was applied to YOLOv4 in the 0015 dataset, it was possible to increase the mAP by 2.19% with an FPS loss of about 3.7%. In the 0015 dataset, in the case of YOLOv3 and YOLOv4, the truck class was not detected. Since BOK is dependent on the detection result from the existing network, the trucks are not detected even with our module applied.

When the rectifying and tracking keys were used individually, the mAP increased. When both keys were used simultaneously, the increase was even greater. When Dataset 0015 and ResNet50 and YOLOv2 were used simultaneously, the mAP significantly improved up to 3.02% compared to the reference. Additionally, the reason the AP of all classes did not rise is because we optimized the target with the highest mAP. Since the movement of a movable object and a static object are different, further improvement is possible in the case of aiming to improve the AP of a specific class.

Fig. 8 is a visualization of the results of applying BOK to various datasets. The red box is the box added by the rectifying key, and the green box is the box added by the tracking key. When compared with the left image to which BOK is not applied, it can be seen that an object with missing detection is additionally detected.

3) HYPERPARAMETER OPTIMIZATION OF THE BUNCH OF KEYS NETWORK

The novel network presented in this paper uses two hyperparameters: the GIoU threshold and CIoU criterion. Assuming that each parameter unit is set between 0 and 1 and the unit step used is 0.01, 10,000 experiments are necessary to determine the optimized parameter for the dataset and network, which takes too much time. Kernel searching was introduced to solve this problem by taking advantage of the fact that it has
the same shape as the peak in Fig. 6. Because the value near the maximum value is also larger than the surrounding values, the maximum value can be found using kernel searching. The optimal value can be found by checking only some of the 100×100 matrices without checking all of them.

The method used in this paper proceeds with two steps. As presented in Fig. 7(a), the matrix value was extracted in a specific unit to determine the global max value. In the experiment, GIoU and CIoU score were composed of 0.25 units from 0 to 1, and a total of 16 values were calculated. As illustrated in Fig. 7(b), the surrounding values are searched for as much as the kernel size set in advance at the point with the highest value from global searching results. After acquiring all data inside the kernel, it moves toward the greatest value. This task repeats until two cases occur:

- four identical values exist in the stored maximum value vector (the maximum value is saved each cycle) and
- the kernel center coordinates move to those saved in the previous cycle.

When the iteration stops, the average values of the previous and current centroid coordinates are returned as the maximum value. The optimal value could be found about 400 times using this algorithm in an experiment that required 10,000 trials. In addition, if the GIoU threshold is set to 100, the BOK does not work. Thus, when extracting the maximum mAP using the BOK, a situation can be detected in which the AP decreases.

Table IV shows the results of our hyperparameter optimization. Both the rectifying key and the tracking key were used. Most of the checked elements were less than 300, which is the result of searching only 3% of the work that checked all 100×100 matrices. When the optimization technique was used, it can be seen that in all cases, higher mAP was recorded than when using the fixed values (GIoU = 0.5, CIoU = 0.5).

V. Conclusion
In this paper, using the BOK network, the performance can be improved only by post-processing connected to the back of the network as a module without changing the existing network structure without additional training. No additional training is required in deep learning, which requires considerable time to train, which is a significant advantage.

The BOK network does not require substantial computation; thus, the mAP can be improved with a slight FPS loss. Therefore, it is suitable for areas requiring a high operating speed, such as the autonomous driving, aviation, and military fields. In addition, because this method has the versatility that it does not depend on the performance of the existing network, the performance of a high-performance network can be further improved.

However, there is an optimization problem because there are two hyperparameters in BOK. In this paper, optimization using kernel search is used, but this method is insufficient to find the perfect value. If the optimal values of hyperparameters can be utilized without tuning, a more complete module can be configured.

We plan to improve the BOK by introducing new keys and use the robot to verify the BOK for an indoor environment in our future work. In addition, we will apply it to the semantic segmentation detector to confirm the performance improvement.

The proposed network is divided into two blocks, and each filter algorithm is applied. In addition, the degree of network performance improvement varies according to the combination of filters. If a new high-performance filter or a study on the application of a filter is conducted and combined with the proposed BOK network, a network that provides higher performance improvement can be created.

REFERENCES
[1] K. Alex, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, pp. 1097-1105, 2012.
[2] S. Karen and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556
[3] L. Min, Q. Chen and S. Yan, "Network in network," 2013, arXiv:1312.4400. [Online]. Available: https://arxiv.org/abs/1312.4400
[4] C. Szegedy W. Liu, Y. Jia, P. Sermanet and S. Reed, "Going deeper with convolutions," in Proc. IEEE conference on computer vision and pattern recognition, Boston, MA, USA, pp. 1-9, 2015.
[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, pp. 770-778, 2016.
[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE conference on computer vision and pattern recognition, Columbus, OH, USA, pp. 580-587, 2014.
[7] R. Girshick, "Fast r-cnn," in Proc. IEEE international conference on computer vision, Santiago, Chile, pp. 1440-1448, 2015.
[8] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 6, pp. 1137-1149, 2015.
[9] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, pp. 779-788, 2016.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. C. Berg, "Ssd: Single shot multibox detector," in Proc. European conference on computer vision, Amsterdam, Netherlands, pp. 21-37, 2016.
[11] H. Rezatofighi, N. Tsoi, J. Gwak and A. Sadeghian, "Generalized intersection over union: A metric and a loss for bounding box regression," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, pp. 658-666, 2019.
[12] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in Proc. AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 12993-13000, 2020.
[13] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE transactions on image processing, vol. 13, no. 4, pp. 600-612, 2004.
[14] L. Zhang, L. Zhang, X. Mou and D. Zhang, "FSIM: A feature similarity index for image quality assessment," IEEE transactions on Image Processing, vol. 20, no. 8, pp. 2378-2386, 2011.
[15] D. Held, S. Thrun and S. Savarese. "Learning to track at 100 fps with deep regression networks," in Proc. European conference on computer vision, Amsterdam, Netherlands, pp. 749-765, 2016.

[16] A. Geiger, P. Lenz and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, pp. 3354-3361, 2012.

[17] J. Redmon and A. Farhadi. "YOLO9000: better, faster, stronger," in Proc. IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, pp. 7263-7271, 2017.

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, arXiv:1804.02767, [Online]. Available: https://arxiv.org/abs/1804.02767

[19] A. Bochkovskiy, C. Wang and H. Liao, “Yolov4: Optimal speed and accuracy of object detection," 2020, arXiv:2004.10934, [Online]. Available: https://arxiv.org/abs/2004.10934

[20] H. Karunasekera, H. Wang and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size," IEEE Access, vol. 7, pp. 104423-104434, 2019.

[21] D. LEE, "One-shot scale and angle estimation for fast visual object tracking," IEEE Access, vol. 7, pp. 55477-55484, 2019.

[22] Y. Shin, H. A. Qadir and I. Balasingham, "Abnormal colon polyp image synthesis using conditional adversarial networks for improved detection performance," IEEE Access, vol. 6, pp. 56007-56017, 2018.

[23] Y. Kim, A. S. Uddin and S. Bae, "Local Augment: Utilizing Local Bias Property of Convolutional Neural Networks for Data Augmentation," IEEE Access, vol. 9 pp. 15191-15199, 2021.

[24] W. Fang, L. Wang and P. Ren, "Tinier-YOLO: A real-time object detection method for constrained environments," IEEE Access, vol. 8, pp. 1935-1944, 2019.

[25] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in Proc. International Conference on Machine Learning, Long Beach, CA, USA, pp. 6105-6114, 2019.