electronics

MDPI

Article

Real-Time Traffic Light Recognition with Lightweight State Recognition and Ratio-Preserving Zero Padding

Jihwan Choi [1] and Harim Lee [2,*]

[1] Department of Semiconductor System Engineering, Kumoh National Institute of Technology, 61, Daehak-ro, Gumi-si 39177, Gyeongsangbuk-do, Republic of Korea; 20236121@kumoh.ac.kr

[2] School of Electronic Engineering, Kumoh National Institute of Technology, 61, Daehak-ro, Gumi-si 39177, Gyeongsangbuk-do, Republic of Korea

* Correspondence: hrlee@kumoh.ac.kr

Abstract: As online shopping is becoming mainstream, driven by the social impact of Coronavirus disease-2019 (COVID-19) as well as the development of Internet services, the demand for autonomous delivery mobile robots is rapidly increasing. This trend has brought the autonomous mobile robot market to a new turning point, with expectations that numerous mobile robots will be driving on roads with traffic. To achieve these expectations, autonomous mobile robots should precisely perceive the situation on roads with traffic. In this paper, we revisit and implement a real-time traffic light recognition system with a proposed lightweight state recognition network and ratio-preserving zero padding, which is a two-stage system consisting of a traffic light detection (TLD) module and a traffic light status recognition (TLSR) module. For the TLSR module, this work proposes a lightweight state recognition network with a small number of weight parameters, because the TLD module needs more weight parameters to find the exact location of traffic lights. Then, the proposed effective and lightweight network architecture is constructed by using skip connection, multifeature maps with different sizes, and kernels of appropriately tuned sizes. Therefore, the network has a negligible impact on the overall processing time and minimal weight parameters while maintaining high performance. We also propose to utilize a ratio-preserving zero padding method for data preprocessing for the TLSR module to enhance recognition accuracy. For the TLD module, extensive evaluations with varying input sizes and backbone network types are conducted, and then appropriate values for those factors are determined, which strikes a balance between detection performance and processing time. Finally, we demonstrate that our traffic light recognition system, utilizing the TLD module's determined parameters, the proposed network architecture for the TLSR module, and the ratio-preserving zero padding method can reliably detect the location and state of traffic lights in real-world videos recorded in Gumi and Deagu, Korea, while maintaining at least 30 frames per second for real-time operation.

Keywords: traffic light recognition; autonomous vehicle; mobile robot; real time; lightweight; deep learning

check for updates

Citation: Choi, J.; Lee, H. Real-Time Traffic Light Recognition with Lightweight State Recognition and Ratio-Preserving Zero Padding. *Electronics* **2024**, *13*, 615. https://doi.org/10.3390/electronics13030615

Academic Editor: Janos Botzheim

Received: 11 December 2023
Revised: 29 January 2024
Accepted: 31 January 2024
Published: 1 February 2024

1. Introduction

In recent years, online transactions have rapidly become a mainstay of the shopping industry. Especially, the Coronavirus disease-2019 (COVID-19) pandemic has significantly impacted shopping behavior, with people preferring to purchase all kinds of essentials, including groceries, food, and clothing, online [1]. Even though the pandemic is almost over, many newcomers to online shopping are expressing continued interest in online shopping [2].

Due to the growth of the e-commerce market, along with the increase in online shoppers, the demand for autonomous delivery mobile robots is rapidly increasing, as well as due to a severe labor shortage. In line with this trend, the autonomous delivery mobile robot market will reach a new turning point, growing at a CAGR of 20.4% from 2021 to

2026 [3]. In addition, various fields require employing autonomous mobile robots [4–6]. Therefore, in the near future, numerous mobile robots are expected to drive on traffic roads to deliver groceries, food, parcels, and other necessities.

In order for the era of autonomous mobile robots to become a reality, it is important for robots to accurately perceive their surroundings on roads with traffic. Specifically, to enable mobile robots to operate autonomously on traffic roads, the real-time identification of traffic lights is of utmost importance.

Several works have been proposed for this purpose, classified into one-stage systems [7–10] and two-stage systems [11–15]. The one-stage system detects the location and state of traffic lights by using a single network. In [7], the authors utilized modified Alexnet networks to analyze detection performance. The works in [8–10] exploited SSD [16], R-CNN [17], and YOLOv3 [18] to detect the location and state of traffic lights. On the other hand, a two-stage system sequentially performs two steps: first finding the location of the traffic light and then checking the signal status of the detected traffic light. In [11], they used a convolutional neural network (CNN) to detect traffic lights, which generated saliency maps. The authors of [13] utilized YOLO [19] to locate traffic lights, with a CNN being used for traffic light status recognition. The works in [12,14] adopted a histogram of oriented gradients (HOG) for the detection of traffic lights and also used a CNN for checking the status of traffic signals. In [15], the proposed system used deep learning networks for all components of the system to detect traffic signs.

However, achieving good recognition performance with a one-stage system can be difficult due to its network structure. In one-stage systems, a network predicts the location and state of traffic lights simultaneously, while two-stage systems consist of two separate networks for each task. The use of a single network to perform two different tasks simultaneously may potentially hinder performance in the one-stage approach. In fact, the work in [8] presented low precision and recall performance for red and yellow signals. Previous works on one-stage systems also have some limitations. Table 1 summarizes the comparison between our work and related studies. As shown in Table 1, the system in [7] can only recognize a limited number of traffic light states, while the works in [9,10] provide limited evaluation results of the traffic light state recognition (TLSR) networks.

Table 1. Comparison between related works and this work for traffic light recognition.

	Approach	Number of Traffic Light States	Performance Metric	Use of CNN	Advanced Features Used
[7]	One-stage	3	Precision, recall	-	-
[8]	One-stage	3	Precision, recall	-	-
[9]	One-stage	5	No detailed evaluation	-	-
[10]	One-stage	3	Precision, recall but limited evaluation	-	-
[11]	Two-stage	2	Precision, recall	Yes	No
[12]	Two-stage	2	Precision, recall	Yes	No
[13]	Two-stage	3	Accuracy	Yes	No
[14]	Two-stage	5	Accuracy	Yes	No
Ours	Two-stage	7	HF score obtained with precision and recall	Yes	FPN [20], skip connection [21]

The two-stage system is composed of two networks, where the first network identifies the location of traffic lights and the second predicts their state. As shown in Table 1, all previous works on the two-stage system [11–14] have built a network that predicts traffic light states using only a basic CNN constructed by sequentially stacking multiple convolutional layers. Recently, however, several advanced network structures have been developed [20,21]. By utilizing these features to design a new network architecture, we can improve recognition performance while reducing the number of weight parameters. Indeed, the TLSR module proposed in this paper utilizes advanced network structures [20,21], leading to improved recognition performance. Additionally, we conducted a comprehensive evaluation of both the proposed model and previous works using a larger number of traffic light states.

In addition, for the network that detects the location of traffic lights, all the mentioned previous studies provide limited evaluations, which do not take into account the trade-off between detection performance and processing time for different input images and backbone network types. The TLD module should provide good detection performance while ensuring real-time operation. As the size of the input image increases, detection performance increases, but processing speed becomes slower. Therefore, extensive evaluation is required to determine the input image size and backbone network type suitable for real-time operation. In this work, we conduct extensive evaluations with varying input image sizes and backbone network types.

In this paper, we revisit the implementation of a real-time traffic light recognition system from a two-stage viewpoint, comprising a traffic light detection (TLD) module and a traffic light state recognition (TLSR) module. The contributions of this work are summarized as follows:

- To improve the recognition performance of the TLSR module and reduce the number of its weight parameters, we propose a lightweight and effective network using advanced features [20,21], while the existing research utilized a simple CNN stacking convolution layers sequentially. Through evaluation, it was confirmed that the proposed network architecture shows superior recognition performance for seven traffic signals compared with the simple CNN used in [11–14] while using fewer weight parameters. The reduction in the weight parameters allows the proposed model to be used on devices with limited resources. Furthermore, through evaluations, this work also shows how the use of advanced features affects the recognition performance.
- Additionally, in order to enhance the TLSR's performance, we propose to utilize a ratio-preserving zero padding (RZP) approach for data preprocessing. This approach maintains the aspect ratio of traffic lights in images, preventing distortions that can arise from scaling.
- By using the determined system parameters for the TLD module and proposed network architecture trained with the RZP for the TLSR module, the real-time traffic light recognition system is demonstrated using actual videos recorded in Gumi and Deagu, Korea. This demonstration shows that the system operates in real time at over 30 FPS while it stably detects traffic lights and identifies the state of detected traffic lights.

The rest of this paper is organized as follows. The design of our system is presented in Section 2, where two essential system modules are outlined: the traffic light detection (TLD) module and the traffic light state recognition (TLSR) module. In Section 3, to propose a lightweight network structure for the TLSR module, we conduct a comprehensive evaluation by comparing the performance across various network architectures, with and without ratio-preserving zero padding, and assess performance metrics such as HF score–confidence curves, average HF scores, trainable parameters, and the number of convolution layers used. For the TLD module, we provide a comprehensive evaluation, assessing its performance over various input image sizes and network types. This analysis utilizes multiple performance metrics such as precision–recall curves, mean average precision, and frames per second. Based on the results, we determine system parameters, such as input image size and network type, and then present the demonstration results. Finally, Section 5 concludes the paper.

2. Two-Stage Traffic Light Recognition System

In this section, we present the implementation of a system designed to detect traffic lights in image frames captured by a camera on autonomous vehicular robots and to determine the state of the detected traffic lights. The system comprises two key modules: a traffic light detection (TLD) module and a traffic light state recognition (TLSR) module. The TLD module is designed to detect the presence and location of traffic lights in images, while the TLSR module determines the state of each detected traffic light. The TLSR module section describes several candidate network structures for the TLSR module, the construction of the output layer, a loss function, and ratio-preserving zero padding. The TLD module section presents the network structure for this module with a description of the loss functions.

Figure 1 shows the block diagram of the entire system, showing what function each module performs. The TLD module receives video frames from a camera or video file, which are then processed in the TLD network to detect the location of traffic lights. The detected traffic lights are cropped from the input frames, and the cropped images are resized by using the ratio-preserving zero padding. Finally, the resized images are passed through the TLSR network to recognize the status of detected traffic lights.

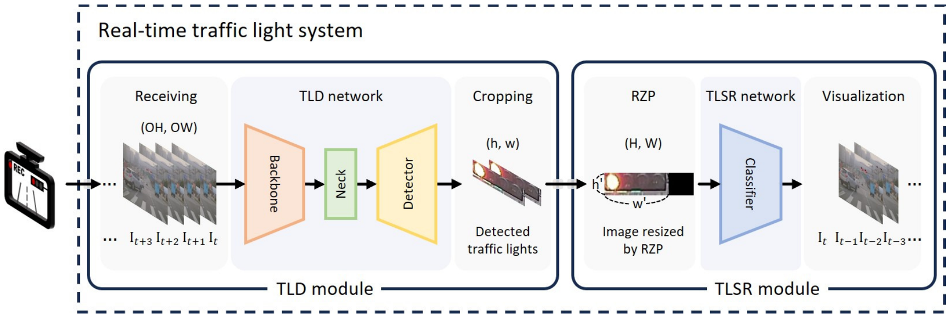


Figure 1. Block diagram of the overall system: the TLD module detects traffic lights and the TLSR module recognizes the status of the detected traffic lights.

2.1. *TLSR: Traffic Light State Recognition*

For the TLSR module, the objective is to design a lightweight and effective recognition network. As mentioned in Section 2, the system in development is a two-stage system in which the TLSR module follows the TLD module. In the two-stage architecture, the TLD module has a complex network structure and numerous weight parameters, which inevitably increases the processing time. Therefore, the processing time of the TLSR module should be much smaller than that of the TLD module to allow the entire system to operate in real time.

In this section, we examine six different network structures, while explaining how to predict the status of the traffic lights and calculate a loss function. These network architectures are constructed by incrementally adding several features in [20,21] and using kernels of appropriately tuned sizes. Furthermore, we describe a ratio-preserving zero-padding method, which enhances the performance of the TLSR module.

2.1.1. Candidate Network Architectures

This section introduces a baseline CNN used in [11–14] and creates five network architectures by leveraging distinct characteristics of [20,21] and adjusting filter sizes in convolution layers. The created structures reduce the number of weight parameters while simultaneously increasing convolution layers by exploiting the distinct features, which improves recognition performance. As previously stated, our aim is to ensure minimal processing time with this secondary module to lessen its impact on the overall processing time of our system. Therefore, in order to reduce the processing time of this TLSR module, the number of weight parameters should be decreased. Furthermore, the network's ability to extract intricate features can be enhanced by increasing the number of convolution layers, which would counteract the degradation caused by reducing weight parameters.

Figure 2 presents the network structures considered: 'CONV', 'CONV-FIT', 'FPN', 'FPN-FIT', 'FPN-RES', and 'FPN-RES-FIT'. The 'Conv' network represents a base structure, built by sequentially stacking convolution layers. The other structures are constructed by integrating supplementary attributes from [20,21] into the base structure. The term 'FPN' refers to utilizing the pyramid structure described in [20], and the networks labeled with

'RES' are designed using the skip connection structure of [21]. For the networks labeled with 'FIT', some convolution layers use filters whose shape corresponds to the height dimension of the incoming feature map.

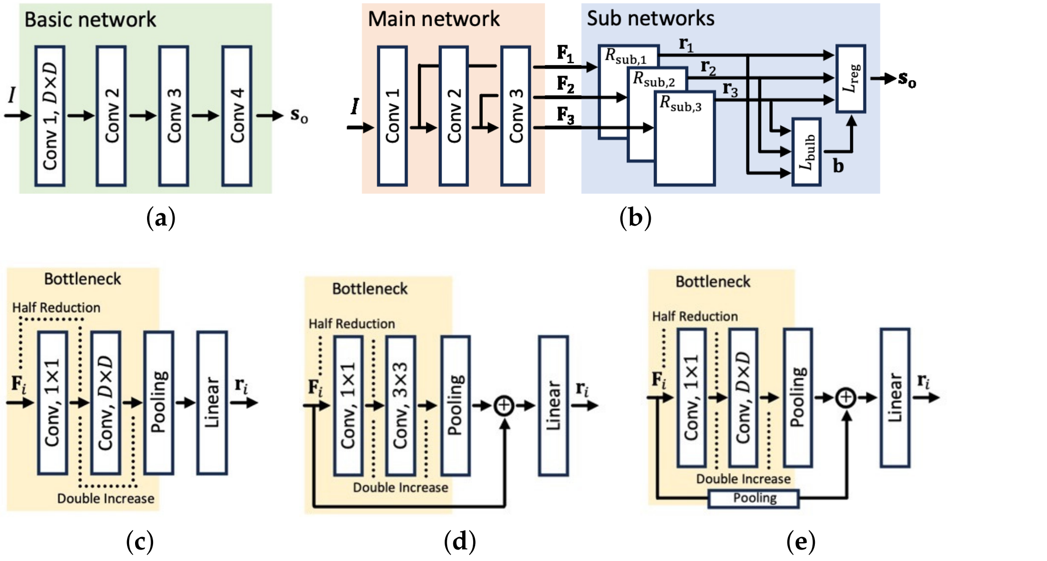


Figure 2. Network architectures for the TLSR module: (**c**–**e**) Subnetwork structures. (**a**) 'CONV' and 'CONV-FIT'; (**b**) 'FPN', 'FPN-FIT', 'FPN-RES', and 'FPN-RES-FIT'; (**c**) For'FPN' and 'FPN-FIT'; (**d**) 'FPN-RES'; (**e**) 'FPN-RES-FIT'.

In Figure 2a, the structure of the 'CONV' and 'CONV-FIT' networks comprises four convolution layers, which is similar to the basic style presented in [22]. In addition, a batch normalization layer and an activation function are followed by each convolution layer, where the Sigmoid Linear Unit (SiLU) function is used for the activation function [23]. For the 'CONV' network, the variable D is set to 3, while set to match the height of the input image I for the 'CONV-FIT' network. This approach could allow the network structure to extract comprehensive features from the input image in a holistic manner.

Figure 2b shows a network design utilizing multiple feature maps with different shapes for predicting the state of traffic signals, which is used for the 'FPN', 'FPN-FIT', 'FPN-RES', and 'FPN-RES-FIT' networks. This structure includes three convolution layers within the main network flow. An output feature map from each main convolution layer is denoted as $\mathbf{F}_i \in \mathbb{R}^{C_i \times W_i \times H_i}$ and $i \in [1, 3]$, where W_i, H_i, and C_i are the width, height, and channel of a feature map, respectively. Each output is passed through a subnetwork as

$$\mathbf{r}_i = R_{\text{sub},i}(\mathbf{F}_i) = L_{\text{sub},i}(C_{\text{sub},i}(\mathbf{F}_i)), \tag{1}$$

where $R_{\text{sub},i}$ represents a subnetwork that follows the i-th main convolution layer, $\mathbf{r}_i \in \mathbb{R}^{N_{\text{reg}} \times 1}$, and N_{reg} is the number of traffic signals. The subnetwork consists of $C_{\text{sub},i}$ and $L_{\text{sub},i}$, where $C_{\text{sub},i}$ is built by stacking two convolution layers, with a 1×1 kernel and a $D \times D$ kernel. Finally, $L_{\text{sub},i}$ has an adaptive average pooling and a linear layer.

By using $\mathbf{r}_i$, the number of bulbs in a traffic light is predicted as

$$\mathbf{b} = L_{\text{bulb}}(\mathbf{r}_{\text{sub}}), \tag{2}$$

where $\mathbf{b} \in \mathbb{R}^{N_{\text{bulb}} \times 1}$, N_{bulb} is the total number of bulbs, L_{bulb} is a classifier consisting of a linear network, and $\mathbf{r}_{\text{sub}} \in \mathbb{R}^{3N_{\text{reg}} \times 1}$ is obtained by concatenating all $\mathbf{r}_i, \forall i$.

Finally, a score vector $\mathbf{s}_o \in \mathbb{R}^{N_{\text{reg}} \times 1}$ is obtained by using $\mathbf{r}_{\text{sub}}$ and $\mathbf{b}$ as

$$\mathbf{s}_o = L_{\text{reg}}(\mathbf{r}_{\text{sub}}, \mathbf{b}), \tag{3}$$

where L_{reg} is a recognizer consisting of a linear network. The bulb vector $\mathbf{b}$ is used as a conditional vector. Since the position of the green bulb in a traffic light can vary depending on N_{bulb}, the condition vector could help L_{reg} accurately predict the traffic signal state.

Figure 2c presents a subnetwork $R_{\text{sub},i}$ used for the 'FPN' network, where the variable D is set to 3. The 1×1 convolution layer reduces the number of channels in $\mathbf{F}_i$ by half. Afterward, the following 3×3 convolution layer increases the number of channels twofold. The 'FPN-FIT' network also employs the subnetwork structure shown in Figure 2c, except that the value of parameter D is adjusted to correspond to the height of $\mathbf{F}_i$.

As shown in Figure 2d, the 'FPN-RES' network uses a subnetwork that adopts the skip connection of [21]. Therefore, $R_{\text{sub},i}$ can be rewritten as

$$\mathbf{r}_i = R_{\text{sub},i}(\mathbf{F}_i) = L_{\text{sub},i}(C_{\text{sub},i}(\mathbf{F}_i) + \mathbf{F}_i). \tag{4}$$

The 'FPN-RES-FIT' network utilizes a subnetwork structure shown in Figure 2e, and the value of parameter D is set to match the height of $\mathbf{F}_i$, similar to the 'FPN-FIT' structure.

For both the 'FPN-RES' and 'FPN-RES-FIT' networks, a score vector is obtained in the same way as in (2) and (3).

2.1.2. Prediction and Loss Function

For each network structure, the final recognizer L_{reg} generates a score vector $\mathbf{s}_{\text{o}} \in \mathbb{R}^{N_{\text{reg}} \times 1}$. In this work, seven traffic signal classes are considered: red, yellow, green, left arrow, left top arrow, left down arrow, and right top arrow (the number of traffic signals may vary depending on the road, city, country, etc.). Hence, each element of $\mathbf{s}_{\text{o}}$ represents the score of each traffic signal.

By passing $\mathbf{s}_{\text{o}}$ through a sigmoid function, the predicted vector $\mathbf{p} \in \mathbb{R}^{N_{\text{reg}} \times 1}$ is calculated, where p_i represents the i-th element of $\mathbf{p}$ and has a value between 0 and 1. In other words, p_i means the probability that the i-th traffic signal is active. As a result, the TLSR module determines that the i-th traffic signal is active if p_i is greater than a certain threshold, and thus can recognize multiple active traffic signals.

In addition, all p_i are below a specific threshold, which indicates that the image received from the TLD module does not contain any traffic lights. Therefore, this TLSR module is capable of filtering out erroneous detection results from the TLD module.

Finally, we compute the total loss E_{total}, which is

$$E_{\text{total}} = E_{\text{reg}} + E_{\text{bulb}}, \tag{5}$$

where E_{reg} is the recognition loss, and E_{bulb} is the classification loss. The loss value E_{reg} is calculated by using the focal loss [24] between $\mathbf{s}_{\text{o}}$ and $\mathbf{t}_{\text{s}}$, which is

$$E_{\text{reg}} = -\sum_{i=1}^{N_{\text{reg}}} [\mathbf{t}_{\text{s}}]_i \cdot \left(1 - \frac{\exp([\mathbf{s}_{\text{o}}]_i)}{\sum_{c=1}^{N_{\text{reg}}} \exp([\mathbf{s}_{\text{o}}]_c)}\right)^{\gamma} \log\left(\frac{\exp([\mathbf{s}_{\text{o}}]_i)}{\sum_{c=1}^{N_{\text{reg}}} \exp([\mathbf{s}_{\text{o}}]_c)}\right), \tag{6}$$

where $\mathbf{t}_{\text{s}}$ is a multi-hot-encoded ground truth vector, $[\mathbf{t}_{\text{s}}]_i$ is the i-th element of $\mathbf{t}_{\text{s}}$, $[\mathbf{s}_{\text{o}}]_i$ is the i-th element of $\mathbf{s}_{\text{o}}$ in (3), and γ is a hyperparameter with $\gamma = 2$. Additionally, the loss value E_{bulb} between $\mathbf{b}$ and $\mathbf{t}_{\text{b}}$ is calculated by using a cross-entropy loss function, which is

$$E_{\text{bulb}} = -\sum_{i=1}^{N_{\text{bulb}}} [\mathbf{t}_{\text{b}}]_i \cdot \log\left(\frac{\exp([\mathbf{b}]_i)}{\sum_{c=1}^{N_{\text{bulb}}} \exp([\mathbf{b}]_c)}\right), \tag{7}$$

where $\mathbf{t}_{\text{b}}$ is a one-hot-encoded ground truth vector, $[\mathbf{t}_{\text{b}}]_i$ is the i-th element of $\mathbf{t}_{\text{b}}$, and $[\mathbf{b}]_i$ is the i-th element of $\mathbf{b}$ in (2).

2.1.3. RZP: Ratio-Preserving Zero Padding

To train a network, input images must be scaled to a fixed size. However, simply scaling an image to fit a specific size can distort objects within the image. In Figure 3, the images in the second and third columns show traffic light images scaled to a fixed size of 20 × 20 pixels and a fixed size of 20 × 100 pixels. For objects with a large width-to-height ratio, such as traffic lights, simply resizing without considering the ratio can distort the position or shape of the light bulb representing the traffic light status. The distortion can degrade the model's prediction performance.

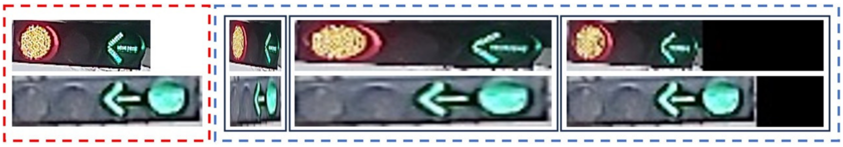


Figure 3. Comparison of various resizing approaches.

In order to improve the performance of the TLSR module, we propose to utilize a ratio-preserving zero padding (RZP) approach to detect traffic light images. With RZP, the width-to-height ratio is preserved so that the position and shape of a bulb do not change after resizing, as shown in the last column image in Figure 3.

To explain RZP, the height and width of a resultant resized image by RZP are H and W. While the value of H is set to a given value, the value of W is determined as $W = H \times R_{\mathrm{w}}$ and $R_{\mathrm{w}} \geq \max(R)$, where R is written as

$$R = \{w/h \mid w = W(D(i)),\ h = H(D(i)),\ i \in [i, N_{\mathrm{D}}]\}, \tag{8}$$

where D represents a dataset used to train the TLSR module, $D(i)$ is the i-th image, N_{D} is the number of images in the dataset D, and $W(\cdot)$ and $H(\cdot)$ are the operators obtaining the width and height of an image $D(i)$. That is, R is the set including the ratio of width to height of all the images in the dataset D.

Using the determined H and W with R_{w} set to a value satisfying $R_{\mathrm{w}} \geq \max(R)$, the width w of a detected traffic light image is resized as

$$w' = w \times \frac{h'}{h}, \tag{9}$$

where h and h' are the height of a detected traffic light image and the resized height, respectively. Specifically, $h' = H$.

Since the size of a detected traffic light image varies, the resized width w' can be smaller than W. If $w' < W$, the remainder $(W - w')$ is padded with zeros so that the resized width w' becomes equal to the determined width W.

2.2. TLD: Traffic Light Detection

Figure 4 shows the overall architecture of the TLD module, consisting of three blocks: a backbone block, a neck block, and a detection block. In the backbone block, a convolutional neural network extracts feature maps and consists of N_{layer} convolution layers, denoted as B_i, $i \in [1, N_{\mathrm{layer}}]$. The output feature map of a layer B_i defines $\mathbf{F}_{(\mathrm{b},i)} \in \mathbb{R}^{(C_i \times H_i \times W_i)}$ and $i \in [1, N_{\mathrm{layer}}]$. The neck block consists of two convolution layers denoted as N_4 and N_3. From a convolution layer N_i, a feature map $\mathbf{F}_{(\mathrm{n},i)}$ is extracted by adding a feature map $\mathbf{F}_{(\mathrm{b},i)}$. To recognize information about traffic lights, the detect block processes each feature map $\mathbf{F}_{(\mathrm{n},i)}$ with an efficient decoupled head block (EDH), denoted as $D_{\mathrm{EDH},i}$. More specifically, the EDH block consists of two parts: anchor-based and anchor-free-based parts. In the anchor-based part (or anchor-free-based part), a classifier head detects the existence of traffic lights $\mathbf{O}^{H \times W \times 3}_{\text{cls, ab}}$ (or $\mathbf{O}^{H \times W \times 1}_{\text{cls, af}}$) while a regressor head predicts the information of bounding boxes of traffic lights $\mathbf{O}^{H \times W \times 12}_{\text{reg, ab}}$ (or $\mathbf{O}^{H \times W \times 68}_{\text{reg, af}}$).

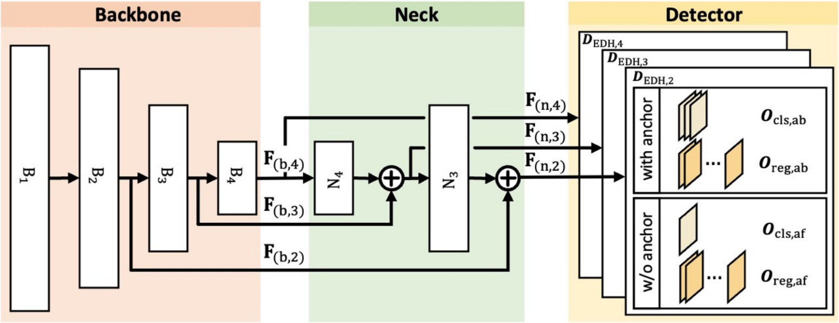


Figure 4. The architecture of the TLD module.

2.2.1. Network Structure

Backbone: A backbone block is built based on [21] and is at the front of the TLD module to extract various pieces of feature information from the input data. It should be able to extract both low-level and high-level complex features from arbitrary input data. There have been many previous works that proposed network structures extracting meaningful features. In [22,25], they propose a method using numerous layers to increase the network depth. To enhance extraction performance, the works in [25–28] propose network blocks suitable for specific applications. In [21,29], the proposed network structures solved gradient vanishing/exploding problems caused by the increasing depth. Additionally, the network structure in [21] is used in various research, and thus, we also adopt it for the backbone network.

Neck: A neck block is constructed based on the feature pyramid network [20]. This block generates new feature maps $\mathbf{F}_{(n,3)}$ and $\mathbf{F}_{(n,2)}$ by using $\mathbf{F}_{(b,3)}$ and $\mathbf{F}_{(b,2)}$ extracted from B_3 and B_2. Then, $\mathbf{F}_{(n,4)}$ is from $\mathbf{F}_{(b,4)}$. For instance, $\mathbf{F}_{(n,3)}$ is obtained by adding $\mathbf{F}_{(b,3)}$ with the result from N_4 where a relatively low-level feature map $\mathbf{F}_{(b,4)}$ is upsampled. In the neck block, the output is calculated as

$$\mathbf{F}_{(n,i)} = C_{\text{lateral}}(\mathbf{F}_{(b,i)}) + N_{(i+1)}(\mathbf{F}_{(b,i+1)}), \tag{10}$$

where $C_{\text{lateral}}(\cdot)$ represents a lateral connection consisting of convolution layers with a 1×1 filter, and $N_i(\cdot)$ is an upsampling function using transposed convolution layers or interpolation functions.

The reason for using multiple feature maps of different sizes is that small objects in an input image can be detected from a low-level feature map. In particular, traffic lights can become smaller depending on the distance between a car and a traffic light, and thus, this method allows better detection of traffic lights of various sizes. In addition, the neck block adds high-level features to low-level features, making low-level features more powerful in representing semantic information.

The backbone network with $N_{\text{layer}} = 4$ generates four feature maps, $\mathbf{F}_{(b,1)}$, $\mathbf{F}_{(b,2)}$, $\mathbf{F}_{(b,3)}$, and $\mathbf{F}_{(b,4)}$ (unlike [20], we remove the convolution layers following $\mathbf{F}_{(b,4)}$ to reduce weight parameters). The neck block utilizes the backbone's feature maps, excluding $\mathbf{F}_{(b,1)}$, since it has too low-level feature information.

Detection: This predicts the bounding boxes of traffic lights using the neck's feature maps, $\mathbf{F}_{(n,i)}$. Since the initial research that pioneered object detection used anchors to predict bounding boxes, subsequent studies have consistently adopted this methodology, called anchor-based detector. In recent years, several works using an anchor-free detector have been introduced, and they are Yolox [30], CornerNet [31], CenterNet [32], and Fcos [33]. Anchor-based object detection places numerous anchor boxes in an image, yet only a small percentage of these anchors actually match the ground truth. The anchor-free object detection solves this inefficiency in anchor-based approaches.

In this paper, the detection block consists of both anchor-based and anchor-free-based detectors, which are trained simultaneously. The anchor-free detector might have difficulty accurately predicting the corner points of bounding boxes in the early stages of training, which can be alleviated by simultaneously training an anchor-based detector. After training, the detection block predicts bounding boxes using only the anchor-free detector.

Each detector is constructed using an efficient decoupled head, consisting of a regression head and a classification head [34,35], as shown in Figure 4. This structure can solve a misalignment between classification confidence and localization accuracy. For an anchor-based detector (an anchor-free-based detector), the classification head predicts the presence or absence of traffic lights in predicted boxes, which is represented by $\mathbf{O}_{\text{cls, ab}} \in \mathbb{R}^{W_i \times H_i \times 3}$ (or $\mathbf{O}_{\text{cls, af}} \in \mathbb{R}^{W_i \times H_i \times 1}$), while the regression head predicts the bounding box information of objects in an image, which is denoted as $\mathbf{O}_{\text{reg, ab}} \in \mathbb{R}^{W_i \times H_i \times 12}$ (or $\mathbf{O}_{\text{reg, af}} \in \mathbb{R}^{W_i \times H_i \times 68}$).

2.2.2. Loss Functions

In order to train the introduced deep learning network, we utilize varifocal loss [36] for classification loss and GIoU loss [37] and DFL [38] for regression loss.

The total loss L_{total} is obtained as

$$L_{\text{total}} = L_{\text{af}} + L_{\text{ab}}, \tag{11}$$

where L_{af} is the anchor-free loss and L_{ab} is the anchor-based loss. The anchor-free loss is calculated as

$$L_{\text{af}} = L_{\text{cls}} + \lambda L_{\text{dfl}}, \tag{12}$$

where L_{cls} and L_{dfl} are classification loss [36] and probability regression loss [38], and λ is a hyperparameter with $\lambda = 2.5$. The anchor-based loss is obtained as

$$L_{\text{ab}} = L_{\text{cls}} + \mu L_{\text{iou}}, \tag{13}$$

where L_{iou} is box regression loss [37], and μ is a hyperparameter with $\mu = 2.5$. As a result, the TLD module is trained using the total loss L_{total} while simultaneously training anchor-based and anchor-free detectors.

2.2.3. Trade-Off between Detection Performance and Processing Time

In order for a TLD module to be applicable in real situations, it must satisfy two main factors, which are high detection performance and low processing time. For detection performance, the module should accurately detect traffic lights of various sizes because the size of a traffic light within a video frame changes in real time with a vehicle approaching the traffic light. Due to the rapid change in the size of traffic lights, the module should also guarantee low processing time.

However, it is generally observed that attempting to reduce processing time can negatively impact detection performance. That is, a trade-off exists between detection performance and processing time. Therefore, system parameters for this module must be carefully selected to balance the trade-off. In contrast to the existing literature, we thoroughly investigate the TLD module in Section 3.

2.3. Overall Algorithm

The procedure of the designed system is demonstrated in Algorithm 1. In a real-world scenario, a companion computer would execute Algorithm 1 each time it receives a video frame from a camera. Here, I_{traffic}, b_{traffic}, and N_{traffic} represent the images of identified traffic lights, the information from the bounding boxes in the identified traffic lights, and the number of I_{traffic}. The P_{rzp} and I_{rzp} notations are the operators of ratio-preserving zero padding and padded traffic light images. Finally, $B(\cdot)$ is a function that draws bounding boxes on v.

Algorithm 1: Traffic light detection and detected object status recognition

```
Input: Traffic light detector $D$; traffic light state recognizer $R$; video frame $v$
Output: Postprocessed video frame $\tilde{v}$
  (1) $D(v) \rightarrow I_{\text{traffic}}, b_{\text{traffic}}, N_{\text{traffic}}$   // Traffic light detection in $v$
  (2) if $N_{\text{traffic}} > 0$ then                    // Traffic lights exist
        for $k \leftarrow 1$ to $N_{\text{traffic}}$ do
          - $P_{\text{rzp}}(I_{\text{traffic},\,k}) \rightarrow I_{\text{rzp},\,k}$   // Ratio-preserving zero padding to $I_{\text{traffic},\,k}$
          - $R(I_{\text{rzp},\,k}) \rightarrow \mathbf{p}_k$          // State probability vector of $I_{\text{rzp},\,k}$
          - $B(v, b_{\text{traffic},k}, \mathbf{p}_k) \rightarrow v$     // Drawing the bounding box and state on $v$
        end for
  (3) Terminate the procedure in a video frame with $v \rightarrow \tilde{v}$.
```

3. Evaluation of TLD and TLSR Modules and Demonstration

To achieve real-time operation, a system should ensure a minimum requirement of 30 frames per second (FPS). It is also essential to optimize the system's detection performance while meeting the minimum FPS requirement. However, there is a trade-off between FPS and detection performance. Therefore, it is crucial to achieve a balance between FPS and detection performance.

For the TLD module, the trade-off arises due to input image size and network depth. Increasing the size of an input image can improve detection performance, but it also results in longer network processing time and consequently reduces FPS. Furthermore, increasing the depth of a network leads to improved detection performance; however, it results in increased processing time.

As a result, the size of an input image and the depth of a backbone network are system parameters for the TLD module. In this section, based on comprehensive evaluations, we determine suitable system parameters for real-time operation.

For the TLSR module, a lightweight and effective network structure is selected from the candidate network structures introduced in Section 2.1.1. All the network architectures are compared in terms of HF score, number of trainable network parameters, and number of convolutional layers. The HF score is a metric introduced in this work, which is a harmonic mean between the F1 score of each traffic signal class and the F1 score of the correctness of a traffic light detected by the TLD module.

3.1. Performance Metrics

In order to evaluate the performance of the TLD module, we utilize the precision–recall curve obtained by varying the confidence threshold, which is the minimum value required to detect the presence of a traffic light. Additionally, the metrics of average precision (AP) and frames per second (FPS) are utilized.

To evaluate the performance of the TLSR module, we present the HF score, which is the harmonic mean of the F1 scores for recognizing the state of each traffic signal class and for filtering out non-traffic-light images. The HF score is calculated as

$$\mathrm{HF}_i = 2 \times \frac{F1_{\text{recog},i} \times F1_{\text{filt}}}{F1_{\text{recog},i} + F1_{\text{filt}}}, \tag{14}$$

where $F1_{\text{recog},i}$ is the F1 score of the i-th traffic light signal, $F1_{\text{filt}}$ is the F1 score for filtering out non-traffic-light images, and $0 \leq \mathrm{HF}_i \leq 1$. As HF_i approaches 1, the TLSR module shows good performance in both recognizing and filtering capabilities.

As stated in Section 2.1.2, the TLSR module can filter out incorrectly cropped images by the TLD module if all elements of $\mathbf{p}$ are below a certain threshold, denoted as Γ_i. When a network is trained to improve its filtering performance, however, its ability to classify the status of each traffic signal may decrease. This is due to how an activated traffic signal is identified, where the i-th traffic signal is predicted as activated if $p_i > \Gamma_i$. Therefore, an evaluation considering both aspects of performance simultaneously is essential due to the trade-off between them. Therefore, by using the HF score, we can concurrently evaluate the candidate network structures by simultaneously considering their ability to filter images without traffic lights and their performance in recognizing traffic light states.

In order to evaluate the efficiency of the TLSR module, we consider two factors: the number of trainable parameters and floating point operations (FLOPs), where FLOPs is a metric that represents the computational complexity of a network.

3.2. Training Details

To train both the TLD and TLSR modules, we utilized the stochastic gradient descent (SGD) optimizer with a weight decay of 5×10^{-4} and a momentum of 0.834, alongside the cosine scheduler. When training the TLD module with the scheduler, we applied an initial learning rate of 3.2×10^{-4} and a final learning rate of 3.84×10^{-5}. Similarly, for the TLSR module, we established an initial learning rate and a final learning rate of 1×10^{-5}. The training and demonstration are conducted utilizing the NVIDIA GeForce RTX 4090. For the TLD module, before training with the traffic light dataset, we perform a pretraining procedure with the coco dataset and the imagenet dataset [39,40].

In order to determine system parameters for the TLD module and a network structure suitable for the TLSR module, both modules are trained with the traffic light dataset provided by the Electronics and Telecommunications Research Institute (ETRI), which is one of the most influential national research institutions in Korea [41]. This dataset consists of 10k images obtained from real-life road environments. The limited dataset allowed quicker testing of various models across various conditions, while also providing enough data for training large networks. This approach is often used for the deep learning field, as in [42]. For training the TLSR module, we intentionally crop the areas lacking traffic lights and utilize them to enhance the TLSR module's ability to filter out the erroneously detected traffic light images by the TLD module.

After selecting suitable parameters and network architecture, for the demonstration, both modules are trained using 230k additional data, collected in Seoul and nearby cities in Korea.

3.3. Network Structure Suitable for TLSR Module

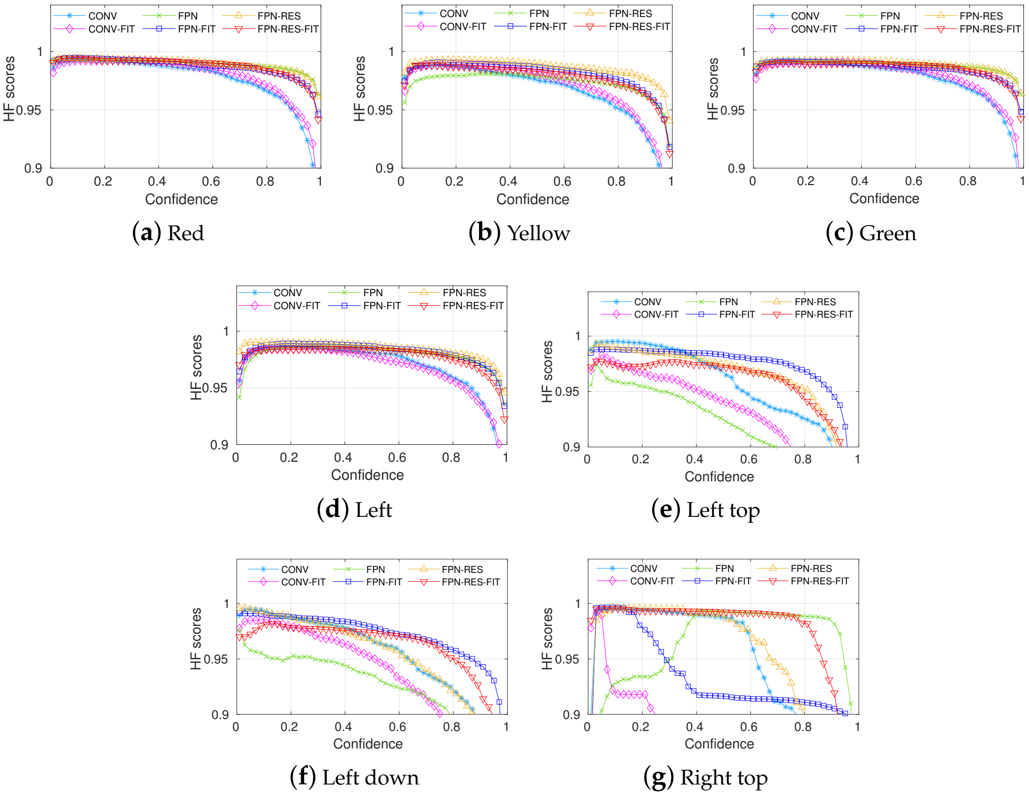
3.3.1. Comparison of HF Scores with Different Network Structures

Figure 5 shows the HF scores of six traffic light signals with different levels of confidence. Note that all the existing work [11–14] on two-stage systems uses the 'CONV' architecture, and thus, 'CONV' can be used as the baseline for the performance comparison.

For the most frequent signals, namely red, yellow, green, and left, it can be observed from Figure 5a–d that the 'CONV' and 'CONV-FIT' network designs show lower HF scores compared with other networks for almost all confidence values. Moreover, it is evident that neither of the network designs outperforms other network architectures.

In Figure 5e, the 'FPN-FIT', 'FPN-RES', and 'FPN-RES-FIT' structures outperform other network architectures in the top-left case, while 'FPN-FIT' and the 'FPN-RES-FIT' are better than the other networks in Figure 5f. For the top-right signal in Figure 5g, the 'FPN' and 'FPN-RES-FIT' architectures exhibit superior performance compared with the others. The findings suggest that the feature pyramid structure in [20] has a significant impact on improving performance. In addition, the 'FPN-RES' architecture shows outstanding performance, as in Figure 5a–d, confirming that the skip connection from [21] also yields significant performance improvements.

Table 2 presents the average of the HF scores across all traffic signals and trainable parameters. The 'FPN-RES-FIT' architecture shows the highest average HF score with 27% fewer weight parameters compared with the 'CONV' network design. Additionally, the computational requirement of the 'FPN-RES-FIT' network structure is 0.30K FLOPs, while that of 'CONV' is 0.39K FLOPs. Because of its small number of weight parameters and low computational complexity, its processing time is negligible. As a result, the 'FPN-RES-FIT' network design is the most appropriate for the TLSR module.



(**a**) Red (**b**) Yellow (**c**) Green

(**d**) Left (**e**) Left top

(**f**) Left down (**g**) Right top

Figure 5. HF scores for traffic light classes.

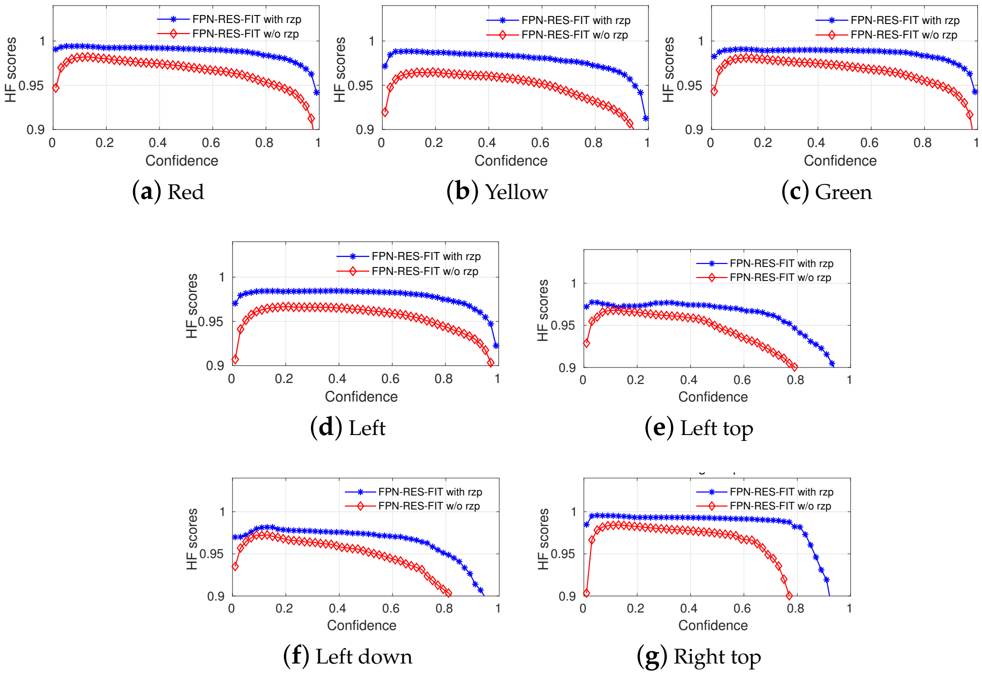
Remark: We need to examine the results of the ‘FPN-RES-FIT’ network structure. The ‘FPN-RES-FIT’ network design shows generally better performance, but it is slightly inferior to the ‘FPN-RES’ and ‘FPN-FIT’ networks in the top left and bottom left of Figure 5e,f. The application of ‘RES’ or ‘FIT’ to ‘FPN’ improves performance. However, applying ‘RES’ and ‘FIT’ simultaneously results in slightly lower performance improvement compared with applying them separately. The reason can be explained as follows. In Figure 2e, in order to apply ‘RES’ and ‘FIT’ simultaneously, a pooling operation should be implemented in the shortcut path. The pooling operation is applied along the width and height dimensions, which reduces spatial information in the resulting feature maps. Due to the relatively higher number of classes associated with the left direction, this reduction may slightly impair the network’s ability to detect the top left and bottom left. In future work, based on this insight, we plan to develop a shortcut block that can effectively combine ‘RES’ and ‘FIT’.

Table 2. Comparison of HF scores, number of trainable parameters, FLOPs, and total number of convolution layers used.

	CONV [11–14]	**CONV-FIT**	**FPN**	**FPN-FIT**	**FPN-RES**	**FPN-RES-FIT**
Average of HF scores	0.9573	0.9205	0.9607	0.9699	0.9708	0.9740
Trainable parameters	98,841	117,609	52,199	72,039	52,199	72,039
FLOPs	0.39 K	0.46 K	0.23 K	0.30 K	0.23 K	0.30 K
Convolution layers	4	4	9	9	9	9

3.3.2. Comparison of HF Scores with and without Ratio-Preserving Zero Padding

This section examines the impact of the RZP method on the ‘FPN-RES-FIT’ network’s performance. Figure 6 presents the HF scores of six traffic signals with and without the RZP method. At all confidence levels, the network with the RZP method results in superior performance compared with the one without it. As a result, maintaining the aspect ratio of traffic lights significantly increases recognition performance.



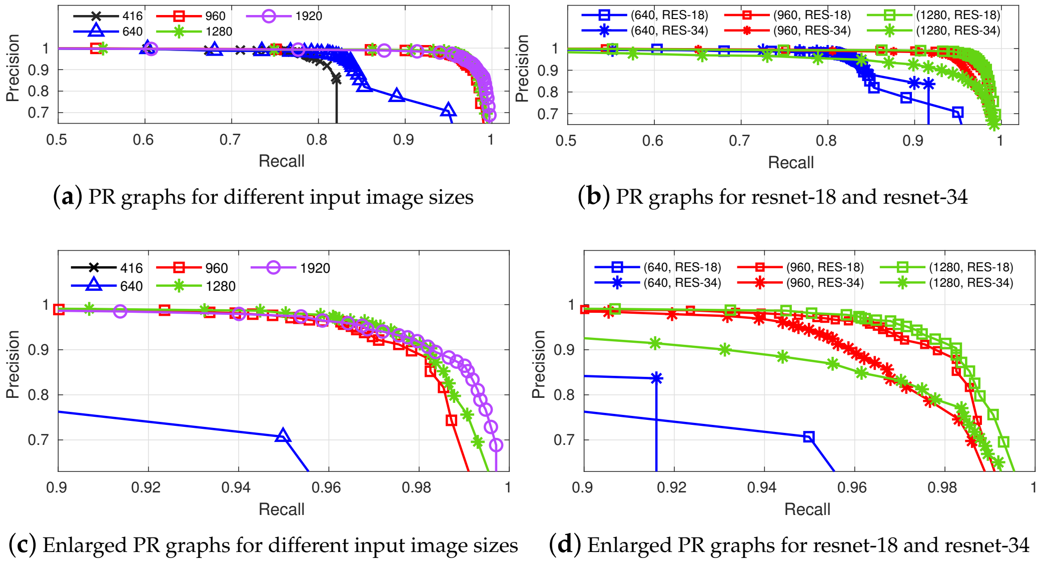
(**a**) Red (**b**) Yellow (**c**) Green

(**d**) Left (**e**) Left top

(**f**) Left down (**g**) Right top

Figure 6. Comparison of HF scores of FPN-RES-FIT with and without ratio-preserving zero padding.

Figure 7 shows the precision–recall curve (PR) of the TLD module with changes in backbone type and input size. When the intersection over union (IoU) threshold of nonmaximum suppression (NMS) is set to 0.5, Figure 7a,b shows the PR curve for 30 confidence thresholds between 0 and 1. Figure 7a presents detection performance for the input size range [416, 640, 960, 1280, 1920], and the backbone type is fixed to resnet-18. Figure 7b shows detection performance for the input size range [640, 960, 1280], and the backbone type is changed from resnet-18 to resnet-34. For a detailed explanation, Figure 7c,d is an enlargement of specific parts of Figure 7a,b. The terms 'RES-18' and 'RES-34' in the legend of the figures signify that the results are obtained with resnet-18 and resnet-34, respectively.



(**a**) PR graphs for different input image sizes (**b**) PR graphs for resnet-18 and resnet-34

(**c**) Enlarged PR graphs for different input image sizes (**d**) Enlarged PR graphs for resnet-18 and resnet-34

Figure 7. Precision–recall curves of the TLD module.

3.4. System Parameters for TLD Module: Input Image Size and Backbone Network Type

3.4.1. Comparison of HF Scores with Different Input Image Sizes

Tables 3 and 4 show the average precision (AP) and FPS. AP is the area under the PR curve, and FPS is the number of frames processed per second by the TLD module. In the tables, AP@0.5 is the result of setting the IoU threshold to 0.5, and AP@0.5:0.95 is the result of averaging APs measured by changing the IoU threshold from 0.5 to 0.95.

Table 3. Comparison of average precision and frame per second for various input image sizes.

Input Image Size	416	640	960	1280	1920
Average precision (AP) @ 0.5	0.8033	0.8825	0.9846	0.9885	0.9913
Average precision (AP) @ 0.5:0.95	0.7499	0.8208	0.9127	0.9115	0.9106
Frames per second (FPS)	143.2	93.3	51.1	30.8	14.5

Table 4. Comparison of average precision and frame per second for various resnet types.

Input Image Size	640		960		1280	
Resnet Type	**18**	**34**	**18**	**34**	**18**	**34**
Average precision (AP) @ 0.5	0.8824	0.8860	0.9850	0.9805	0.9884	0.9160
Average precision (AP) @ 0.5:0.95	0.8202	0.8278	0.9127	0.8865	0.9115	0.8413
Frame per second (FPS)	93.3	49.6	51.1	26.4	30.8	16.6

In Figure 7a–d, for image sizes from 960 to 1920, the area of the PR curve is almost 1, while the area for 416 and 640 is smaller. In Tables 3 and 4, the results of AP@0.5 show that the detection performance becomes higher as the input size increases. On the other hand, for AP@0.5:0.95, the detection performance increases up to an input size of 960 and then decreases. For FPS, Table 3 shows that it decreases as the input size increases. Especially, for input sizes larger than 1280, FPS is fewer than 30, which is the minimum requirement for real-time operation. Table 4 confirms that the processing speed of resnet-18 is 46.7%, 48.3%, and 46.1% faster than that of resnet-32 for input sizes of 640, 960, and 1280, respectively.

Finally, Figure 8 shows the resulting images for the input sizes 640 and 960. Small traffic lights within the same image are better detected with an input image size of 960 than with an image size of 640. This explains that the larger the image, the better the detection performance. Furthermore, the same holds true when there are multiple traffic lights.

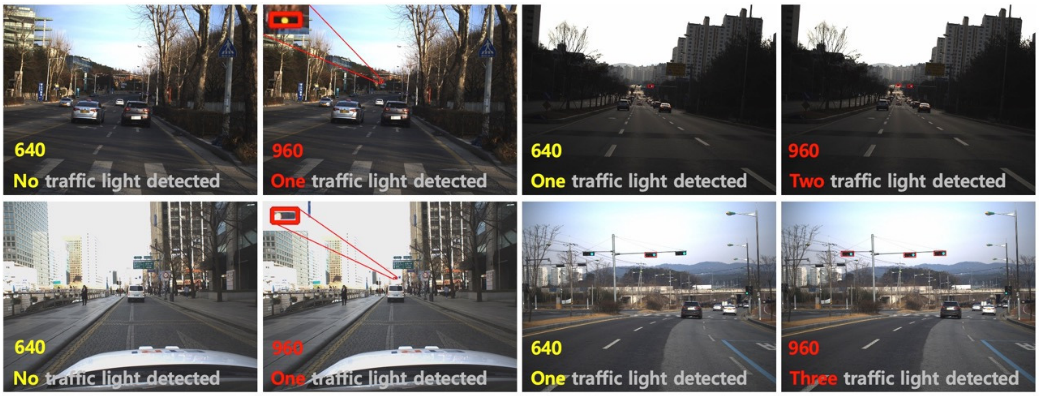

640
No traffic light detected
960
One traffic light detected
640
One traffic light detected
960
Two traffic light detected
640
No traffic light detected
960
One traffic light detected
640
One traffic light detected
960
Three traffic light detected

Figure 8. Example showing the difference in performance between 640 and 960 input image sizes.

As a result, extensive evaluations confirm that there is a trade-off between detection performance and processing time. Therefore, since the minimum requirement for real-time operation is typically 30 FPS, we decided to set the input size and backbone type to 960 and resnet-18, respectively, for the TLD module.

3.4.2. Remark

In this paper, we focus on the investigation of the trade-off between detection performance and processing time for the TLD module. In particular, the TLD module is constructed following YOLOv5. The recent work in [15] also uses YOLOv5 to detect traffic signs that are different from traffic lights. Although it is a study to find different objects, it shows that YOLOv5 outperforms YOLOv3 and v4 when detecting objects in a driving environment. Furthermore, through our extensive evaluation, we can confirm that the average precision of the TLD module is almost 1 for input image sizes of 960, 1280, and 1920 for the ETRI dataset. Therefore, we believe it is appropriate to evaluate the trade-off only for the TLD module we established.

3.5. Demonstration

Finally, we demonstrate our real-time traffic light recognition system through videos recorded on various traffic roads in Gumi and Daegu, Korea, which are not the cities where the dataset was collected. Based on evaluations, the TLD module utilizes an input image size of 960 and a backbone network type of resnet-18, while the 'FPN-RES-FIT' network trained with RZP is used for the TLSR module. This setting provides the best performance while ensuring the minimum requirement of 30 FPS (if the input image size is 1280, the FPS for the whole procedure are fewer than 30). Finally, the proposed system for this demonstration runs on a Geforce RTX 4090, with 2.06 MB memory for the TLSR module and 2.317 GB memory for the TLD module.

Figure 9 shows several resulting video frames, including the bounding boxes and states of detected traffic lights, and also records the FPS of the two-stage system. Additionally, the location of the traffic road was marked on Google Maps and displayed in the lower left corner. Through the resulting images, it was confirmed that traffic lights of various angles and sizes can be detected and that various types of traffic lights can be recognized in an actual car driving environment. More video results can be found at the following link: https://youtu.be/mQA8THT7OmE?si=-chjWc8h9_zc6Xuk, accessed on 30 January 2024.

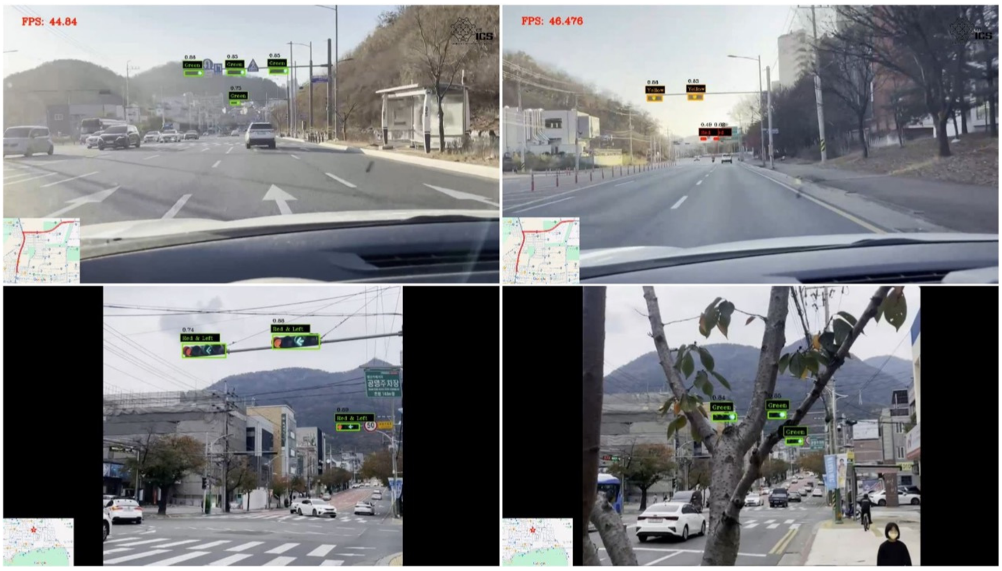


Figure 9. Demonstration images using videos taken on various domestic traffic roads.

4. Discussion

4.1. Necessity of Real-Time Traffic Light Recognition System

We can think about how vehicles use big data from traffic lights. Two cases are considered: (1) when big data are on remote servers and provided to vehicles through wireless communication and (2) when big data are imported into vehicles.

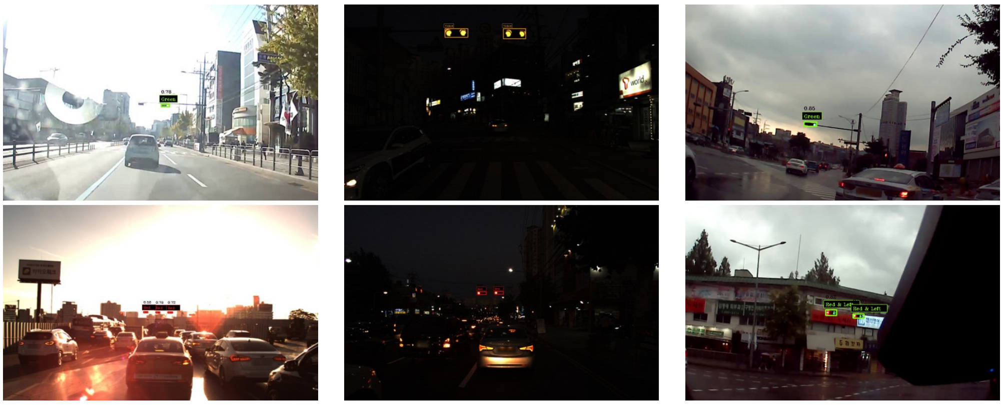
For the first situation, to recognize the status of the traffic lights in front of vehicles, they need to be connected to a remote server using vehicle communication networks such as VANET (Vehicular Ad hoc Network) or infrastructure networks such as LTE and 5G, where networks are established using wireless communication. However, the performance of wireless communication, such as throughput and delay, can vary. As the number of vehicles using wireless communication increases, the throughput decreases and the delay increases. Therefore, when wireless communication performance is poor, the vehicle has to determine the status of the traffic lights ahead without information from the server.

For the second scenario, vehicles can identify the status of traffic lights by utilizing the big data stored in them. However, they would not be able to handle changing situations, such as the dynamic variation in traffic light flashing patterns. In Korea, to maintain smooth traffic flow, traffic light flashing patterns can dynamically vary based on real-time traffic conditions, such as congestion. This is because it is difficult to include all cases of traffic flow that varies and changes differently in every situation, making real-time management challenging with imported data alone.

As a result, there is a need to develop technology that can be implemented in vehicles and that allows vehicles to independently determine the status of the traffic light ahead.

4.2. Evaluation under Different Weather Conditions

In the field of computer vision, performance can be affected by weather conditions, including brightness, darkness, cloudiness, and rain. Figure 10 shows that the system developed in this work is capable of detecting and recognizing the location and state of traffic lights in various conditions, such as during sunset and nighttime, in rainy weather, and under cloudy skies. The results present that the system can also operate in different lighting conditions. Unfortunately, however, the training dataset used does not have labels for weather conditions, and thus, it is difficult to determine the amount of data for each weather condition. A brief check shows that there are relatively little data for the above weather conditions. In future work, we plan to conduct research on various weather conditions by collecting additional data and addressing imbalances between them.



(**a**) Brightness (**b**) Darkness (**c**) Cloudiness and rain

Figure 10. Resulting images for several weather conditions: (**a**) brightness, (**b**) darkness, and (**c**) cloudiness and rain.

4.3. Why Should a YOLO-Style Network Be Used for the TLD Module?

In the field of object detection, there are two approaches: one-stage detectors and two-stage detectors. Although two-stage detectors show better performance than one-stage detectors, their slower processing speeds make them unsuitable for real-time systems. Additionally, a new technique called vision transformer has recently been proposed for image classification, which uses transformer architecture. However, it also has limitations, such as high processing time and high memory requirements due to the large number of weight parameters and high computational complexity. Therefore, it is not yet suitable for real-time object detection systems.

The YOLO neural network family is a representative one-stage detector, while the Faster R-CNN [43] is a representative two-stage detector. Specifically, the YOLO neural network family is known for its exceptional balance between accuracy and speed, allowing for the rapid and dependable identification of objects in images. Therefore, for this work, we utilize a YOLO-style network for real-time traffic light detection.

4.4. Why Is 30 FPS Considered the Real-Time Operation Criterion in This Work?

In Korea, the speed limit for vehicles is 50 km/h in urban areas and 30 km/h in school zones. These speed limits translate to 13.8 m/s and 8.3 m/s, respectively. Based on 30 FPS, the elapsed time of 5 frames is 0.16 s, and the vehicle's moving distances during only 5 frames are 2.2 m and 1.3 m, respectively. If the FPS decreases, the moving distance increases, which results in longer reaction times for vehicles. Longer reaction times could increase the risk of accidents. Additionally, the basic specification of commercial cameras is typically 30 FPS. Therefore, this work considers 30 FPS for real-time operation.

5. Conclusions and Future Works

In this paper, we revisit the real-time traffic light recognition system, which is a two-stage system comprising TLD and TLSR modules. Then, we propose a lightweight and effective network architecture for the TLSR module, introduce a data preprocessing approach to improve TLSR performance, and at the same time, determine the suitable system parameters of the TLD module through a comprehensive evaluation.

For the TLSR module, our objective is to construct an effective and lightweight network that has little impact on the overall FPS of the two-stage system. We evaluated various network structures and proposed a network design that uses skip connections, multiple feature maps, and filter sizes customized to the size of the incoming feature maps. The proposed network design shows an average HF score close to 1 while using a few weight parameters. In addition, we introduce the ratio-preserving zero padding approach in traffic light images detected by the TLD module, which improves the recognition performance of the TLSR module.

In the TLD module, to achieve a balance between detection performance and FPS for real-time operation, we conducted a comprehensive evaluation that varied system parameters such as the input image's size and the backbone network type. We compared precision–recall curves, average precision, and FPS to determine a suitable pair that guarantees at least 30 FPS with adequate detection performance.

Finally, we demonstrate a system constructed using the proposed network structure trained with determined system parameters and ratio-preserving zero padding. This demonstration shows that our real-time traffic light detection system can reliably detect traffic lights. Especially, when the vehicle is close to a traffic light, the system can continuously detect the traffic light and accurately identify the status of the detected traffic signal, proving the reliability of the system.

For future work, we plan to improve our real-time system to enable the continuous detection of traffic lights over considerable distances. Our system is also good at finding distant traffic lights but tends to detect them relatively discontinuously compared with nearby traffic lights. Therefore, we will develop a TLD module that considers multiple consecutive video frames as input data. Through this, the TLD module is expected to focus on areas where traffic lights exist in multiple video frames, rather than operating independently for each frame. Additionally, for the development of these TLD modules, we plan to consider one of the recent object detection networks, such as YOLOv8.

Author Contributions: Conceptualization, H.L. and J.C.; methodology, H.L.; software, H.L. and J.C.; validation, H.L.; resources, H.L.; writing—original draft preparation, J.C.; writing—review and editing, H.L.; supervision, H.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No.2022R1G1A1007058), and it was supported by the Institute for Information & communications Technology Promotion(IITP) grant funded by the Korean government (MSIP) (No. 2017-0-00068, A Development of Driving Decision Engine for Autonomous Driving (4th) using Driving Experience Information/2018-0-00327, Development of Fully Autonomous Driving Navigation AI Technology in high-precision map shadow environment), and used datasets from 'The Open AI Dataset Project (AI-Hub, S. Korea)'. All data information can be accessed through 'AI-Hub (www.aihub.or.kr, accessed on 30 January 2024)'.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TLD	Traffic light detection
TLSR	Traffic light state recognition
RZP	Ratio-preserving zero padding
FPS	Frames per second
FLOPs	Floating point operations

The following mathematical symbols are frequently used in this manuscript:

$\mathbf{I}_\mathrm{t}$	The input frame received from a camera or video
$(\mathbf{OH}, \mathbf{OW})$	The height and width of original image
$(\mathbf{h}, \mathbf{w})$	The height and width of detected traffic light
$(\mathbf{H}, \mathbf{W})$	The height and width of image resized by RZP
$(\mathbf{h}', \mathbf{w}')$	Resized height and width of detected traffic lights
$\mathbf{F}_i$	An output feature map from each main convolution layer
$\mathbf{r}_i$	An output of subnetwork
$\mathbf{b}$	The number of bulbs in a traffic light
$\mathbf{s}_\mathrm{o}$	A score vector of TLSR module
B_i	The convolution layers in the backbone block of TLD module
$\mathbf{F}_{(\mathrm{b},i)}$	The output feature map of a layer B_i
N_i	The convolution layers in the neck block of TLD module
$\mathbf{F}_{(\mathrm{n},i)}$	The output feature map of a layer N_i
$D_{\mathrm{EDH},i}$	An efficient decoupled head block
$\mathbf{O}_{\mathrm{cls,\ ab}}$ (or $\mathbf{O}_{\mathrm{cls,\ af}}$)	The output of a classifier head, which is the existence of traffic lights
$\mathbf{O}_{\mathrm{reg,\ ab}}$ (or $\mathbf{O}_{\mathrm{reg,\ af}}$)	The output of a regressor head, which is the information of bounding boxes of traffic lights

References

1. Fokina, M. Online Shopping Statistics: Ecommerce Trends for 2023. 2023. Available online: https://www.tidio.com/blog/online-shopping-statistics/ (accessed on 30 January 2024).
2. Shaw, N.; Eschenbrenner, B.; Baier, D. Online shopping continuance after COVID-19: A comparison of Canada, Germany and the United States. *J. Retail. Consum. Serv.* **2022**, *69*, 103100. [CrossRef]
3. Facts & Factors. *Global Autonomous Delivery Robots Market Size (2020–2026) Share, Industry Trends, Growth, Challenges, and Forecast*; Facts & Factors: Pune, India, 2023
4. Fountas, S.; Mylonas, N.; Malounas, I.; Rodias, E.; Hellmann Santos, C.; Pekkeriet, E. Agricultural Robotics for Field Operations. *Sensors* **2020**, *20*, 2672. [CrossRef] [PubMed]
5. Gonzalez-de Santos, P.; Fernández, R.; Sepúlveda, D.; Navas, E.; Emmi, L.; Armada, M. Field Robots for Intelligent Farms—Inhering Features from Industry. *Agronomy* **2020**, *10*, 1638. [CrossRef]
6. Hajduk, M.; Koukolova, L. Trends in Industrial and Service Robot Application. *Appl. Mech. Mater.* **2015**, *791*, 161–165. [CrossRef]
7. Weber, M.; Wolf, P.; Zöllner, J.M. DeepTLR: A single deep convolutional network for detection and classification of traffic lights. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 19–22 June 2016; pp. 342–348. [CrossRef]
8. Kim, J.; Cho, H.; Hwangbo, M.; Choi, J.; Canny, J.; Kwon, Y.P. Deep Traffic Light Detection for Self-driving Cars from a Large-scale Dataset. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 280–285. [CrossRef]

9. Kulkarni, R.; Dhavalikar, S.; Bangar, S. Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning. In Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018; pp. 1–4. [CrossRef]
10. Possatti, L.C.; Guidolini, R.; Cardoso, V.B.; Berriel, R.F.; Paixão, T.M.; Badue, C.; De Souza, A.F.; Oliveira-Santos, T. Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8. [CrossRef]
11. John, V.; Yoneda, K.; Liu, Z.; Mita, S. Saliency Map Generation by the Convolutional Neural Network for Real-Time Traffic Light Detection Using Template Matching. *IEEE Trans. Comput. Imaging* **2015**, *1*, 159–173. [CrossRef]
12. Saini, S.; Nikhil, S.; Konda, K.R.; Bharadwaj, H.S.; Ganeshan, N. An efficient vision-based traffic light detection and state recognition for autonomous vehicles. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 606–611. [CrossRef]
13. Behrendt, K.; Novak, L.; Botros, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1370–1377. [CrossRef]
14. Ouyang, Z.; Niu, J.; Liu, Y.; Guizani, M. Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles. *IEEE Trans. Mob. Comput.* **2020**, *19*, 300–313. [CrossRef]
15. Kim, C.I.; Park, J.; Park, Y.; Jung, W.; Lim, Y.S. Deep Learning-Based Real-Time Traffic Sign Recognition System for Urban Environments. *Infrastructures* **2023**, *8*, 20. [CrossRef]
16. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2015.
17. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
18. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
19. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: Washington, DC, USA, 2016; pp. 779–788. [CrossRef]
20. Lin, T.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 936–944. [CrossRef]
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: Washington, DC, USA, 2016; pp. 770–778. [CrossRef]
22. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
23. Elfwing, S.; Uchibe, E.; Doya, K. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning. *Neural Netw.* **2018**, *107*, 3–11. [CrossRef] [PubMed]
24. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
26. Newell, A.; Yang, K.; Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2016.
27. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
28. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA., 2017; pp. 1800–1807. [CrossRef]
29. Huang, G.; Liu, Z.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
30. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv* **2021**, arXiv:2107.08430.
31. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. *Int. J. Comput. Vis.* **2020**, *128*, 642–656. [CrossRef]
32. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Republic of Korea, 27 October–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 6568–6577. [CrossRef]
33. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully Convolutional One-Stage Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Republic of Korea, 27 October–2 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 9626–9635. [CrossRef]

34. Song, G.; Liu, Y.; Wang, X. Revisiting the Sibling Head in Object Detector. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; Computer Vision Foundation/IEEE: Piscataway, NJ, USA, 2020; pp. 11560–11569. [CrossRef]
35. Wu, Y.; Chen, Y.; Yuan, L.; Liu, Z.; Wang, L.; Li, H.; Fu, Y. Rethinking Classification and Localization for Object Detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020; Computer Vision Foundation/IEEE: Piscataway, NJ, USA, 2020; pp. 10183–10192. [CrossRef]
36. Zhang, H.; Wang, Y.; Dayoub, F.; Sünderhauf, N. VarifocalNet: An IoU-Aware Dense Object Detector. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, 19–25 June 2021; Computer Vision Foundation/IEEE: Piscataway, NJ, USA, 2021; pp. 8514–8523. [CrossRef]
37. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.D.; Savarese, S. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; Computer Vision Foundation/IEEE: Piscataway, NJ, USA, 2019; pp. 658–666. [CrossRef]
38. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020.
39. Lin, T.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the Computer Vision-ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014.
40. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
41. Electronics and Telecommunications Research Institute (ETRI). ETRI Traffic Light Dataset. 2019. Available online: https://nanum.etri.re.kr/share/kimjy/etri_traffic_light?lang=ko_KR (accessed on 30 January 2024.)
42. Piergiovanni, A.J.; Ryoo, M.S. Representation Flow for Action Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; Computer Vision Foundation/IEEE: Piscataway, NJ, USA, 2019; pp. 9945–9953. [CrossRef]
43. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.