Deep Convolutional Neural Network for Enhancing Traffic Sign Recognition developed on Yolo V4

Christine Dewi [1,2], Rung-Ching Chen [1, *], Xiaoyi Jiang [3], and Hui Yu[4]

Received XXX; revised XXX

Abstract

Traffic sign detection (TSD) is a key issue for smart vehicles. Traffic sign recognition (TSR) contributes beneficial information, including directions and alerts for advanced driver assistance systems (ADAS) and Cooperative Intelligent Transport Systems (CITS). Traffic signs are tough to detect in practical autonomous driving scenes using an extremely accurate real-time approach. Object detection methods such as Yolo V4 and Yolo V4-tiny consolidated with Spatial Pyramid Pooling (SPP) are analyzed in this paper. This work evaluates the importance of the SPP principle in boosting the performance of Yolo V4 and Yolo V4-tiny backbone networks in extracting features and learning object features more effectively. Both models are measured and compared with crucial measurement parameters, including mean average precision (*mAP*), working area size, detection time, and billion floating-point number (BFLOPS). Experiments show that Yolo V4_1 (with SPP) outperforms the state-of-the-art schemes, achieving 99.4% accuracy in our experiments, along with the best total BFLOPS (127.26) and *mAP* (99.32%). In contrast with earlier studies, the Yolo V3 SPP training process only receives 98.99 % accuracy for *mAP* with *IoU* 90.09. The training *mAP* rises by 0.44% with Yolo V4_1 (*mAP* 99.32%) in our experiment. Further, SPP can enhance the achievement of all models in the experiment.

Keywords: Yolo V4, Yolo V3, Spatial Pyramid Pooling, Object Recognition, CNN.

1. Introduction.

Proper preservation of an inventory of traffic signs is an important activity for maintaining traffic quality and productivity (Balali, Ashouri Rad, & Golparvar-Fard, 2015). Traffic signs are collected by employing a vehicle-mounted camera, and identification is performed offline by an operator to examine compatibility with the existing database. However, this manual task is time-consuming. Thus, the discovery and detection of a vast number of groups of traffic signs remains an open topic. Recently, the deep convolution neural network (CNN) has attracted tremendous attention [2, 3]. Previous benchmarks discussed the role of traffic sign identification and detection [4, 5, 6].

Nonetheless, some of these are based solely on traffic sign recognition (TSR) and neglected the more complicated traffic sign-detection (TSD) issue. Recently, TSR has contributed valuable information such as directions and alerts for autonomous driving and driver assistance systems. Further, TSD has acquired other observations from navigation systems for smart vehicles and Cooperative Intelligent Transport Systems (CITS) [7, 8].

* Rung-Ching Chen
✉ crching@cyut.edu.tw
[1] Department of Information Management, Chaoyang University of Technology, Taichung, Taiwan.
[2] Faculty of Information Technology, Satya Wacana Christian University, Central Java, Indonesia.
[3] Department of Mathematics and Computer Science, University of Münster, D-48149 Münster, Germany.
[4] School of Creative Technologies, the University of Portsmouth, UK.

Traffic signs are meant to be known and precise, with a basic design and standard colors. Yet, the identification of traffic signs implies several problems due to differences in sign design between countries. These differences are easy for humans to identify, but remain a great challenge for automated detection systems. Furthermore, developing a robust TSRS system is an essential issue, because of the latency period in the testing time. In this research we focus on Taiwan's prohibitory sign identifications. Our motivation is the lack of a database or research system for Taiwan's traffic sign recognition. We also improve the previous research performance (Dewi, Chen, & Tai, 2020) for the Taiwan Traffic Sign Dataset (TWTSD).

A traffic sign prediction system in which the location and the accurate border of traffic signs are simultaneously predicted using a single CNN [10, 11] is implemented in this study. The latest innovations in CNN You Only Look Once (Yolo) V4 are the foundation of our novel object detection network. Yolo V4 (Bochkovskiy, Wang, & Mark Liao, 2020) has the following benefits: (1) Yolo V4 is a powerful and efficient object detection platform that helps anyone with a 1080 Ti or 2080 Ti GPU to train a super-fast, accurate object detector. (2) The value of state-of-the-art object detection Bag-of-Freebies (BoF) and Bag-of-Specials (BoS) methods has been developed during detector training. (3) Adjusted state-of-the-art techniques, including Cross-iteration batch normalization (CBN) and Path aggregation network (PAN), are now more effective and fit for individual GPU training. Further, based on all the advantages of Yolo V4, this research uses the object identification method.

This article's contributions are (1) Offers a brief discussion of CNN-based object detection algorithms, especially the new Yolo V4 and Yolo V4-tiny versions. (2) Reviews and assesses several state-of-the-art object detectors, notably for the detection of traffic signs. Performance metrics include vital parameters such as *mAP*, detection time, *IoU*, and BFLOPS. (3) This research applies SPP to collect the local region features at different scales in the same CNN layer for learning multi-scale object features more comprehensively. Our findings demonstrate that Yolo V4_1 with SPP is the most accurate model in the experiment. In this study, Yolo V4_1 outperforms state-of-the-art schemes, obtaining 99.4% accuracy in recognition of the Taiwan Traffic Sign Dataset (TWTSD). On average, Yolo V4_1 is 0.4% higher than state-of-the-art Yolo V3 SPP (Dewi et al., 2020). In previous research, Yolo V3 SPP (Dewi et al., 2020) training process obtained only 98.99 % accuracy for mAP with *IoU* 90.09. Thus, accuracy increases by 0.44% with Yolo V4_1 (*mAP* 99.32%).

The remainder of this paper is arranged as follows. Section 2 examines the materials and methods. Section 3 summarizes the findings of the experiment. Section 4 discusses and describes the results obtained for the recognition and analysis of traffic signs. Section 5 discusses the conclusions of the research paper and future research.

2. Related Works.

2.1 Object recognition with CNN

Recently, the development of big data and advanced computing functions like single shot detectors (SSD) (Min, Li, Wang, Zeng, & Liao, 2019), Faster Region-based Convolutional Neural Networks (Faster R-CNN) (Ren, He, Girshick, & Sun, 2017), and Yolo (Redmon & Farhadi, 2017) has gained substantial attention in the field of deep learning methods. Initially, CNN was deepened and expanded for better precision. Recently, the network structure has been made shorter and more successful. CNN that can be trained without handmade features is in widespread use at present. In (Ciregan, Meier, & Schmidhuber, 2012), the classification of traffic signs was suggested for multi-column CNN, in which multiple CNN is trained with different weight initializations or pre-processing data. The Yolo series algorithms (Redmon & Farhadi, 2017) have a typical one-stage network structure. This network structure is more concise than the two-stage R-CNN algorithm network.

Yolo's detection speed is faster than the R-CNN series because the candidate area mechanism and detection are combined into the same network. Though CNN and Yolo have shown excellent achievement in identifying, analyzing, and classify images, building a thriving infrastructure in the network, and establishing a viable model remain challenging tasks.

In the previous research study (Guo, Qian, & Shi, 2021), a real-time railroad track components inspection framework based on the just-released YOLOv4 is proposed for track components inspection quickly and efficiently. The cutting-edge convolutional neural network, YOLOv4 is improved trained, and evaluated based on the images in a public track components image database. Moreover, Tai, S. K., et al (Tai et al., 2020) follow the principle of SPP to strengthen the original structure of Yolo V3 for building features extraction and determining maximum information flow between layers in the system. The result suggests that the detection time would be quicker if a big number of scales is used. Therefore, the precision declines relative to the original Yolo V3 scale. Similarly, in Yolo V3 SPP, the accuracy declines to take on various sizes. In (Avramović et al., 2020) proposes speed and accuracy improvement of traffic sign detection and recognition in high-definition images, based on focusing on different regions of interest in traffic images. These regions are determined with efficient and parallelized preprocessing of every traffic image, after which convolutional neural network is applied for detection and recognition in parallel on graphics processing units.

Our research focus on Taiwan Traffic Sign Dataset (TWTSD). We offer a brief discussion of CNN-based object detection algorithms, especially the new Yolo V4 and Yolo V4-tiny versions. Also, it reviews and assesses several state-of-the-art object detectors, notably for the detection of traffic signs. Performance metrics include vital parameters such as *mAP*, detection time, *IoU*, and BFLOPS. This research applies SPP to collect the local region features at different scales in the same CNN layer for learning multi-scale object features more comprehensively.

Moreover, Yolo V3 (Redmon & Farhadi, 2017) employed Darknet-53 as the backbone network to replace Darknet-19 and used multi-scale estimation. [9, 18] uses Spatial Pyramid Pooling (SPP) combined with Yolo V3 to detect Taiwan prohibitory signs. Most of the latest scientific models require many GPUs for training with a large mini-batch size. Usually, training with one GPU makes the training process slow, heavy, and ineffective. Yolo V4 (Bochkovskiy et al., 2020) addresses this problem by constructing an object detector trained on a single GPU with a smaller mini-batch size. This method makes gives it the potential train a super quick and precise object detector with a single 1080 Ti or 2080 Ti GPU.

2.2 Spatial Pyramid Pooling (SPP) Network

In conventional network architectures, a network usually requires a fixed-size input [20, 21]. This condition is only set for completely connected layers since they require fixed-length vectors. The convolutional layers can prepare random sizes of images and generate their outputs with a similar aspect ratio. Relying on this feature, SPP [22, 23] replaces the latest pooling layer and produces fixed-size vectors. This outcome is transferred to the entirely associated layer. The result of the latest convolutional function map is distributed into space containers of equal sizes in the SPP layer. The number of bins is fixed regardless of the size of the image. Figure 1 presents the network with SPP.

SPP has the following characteristics [24, 25]: (1) For each input dimension, SPP generates a fixed-length element. (2) SPP applies multi-stage pooling to control deformations effectively. (3) Because the input is arbitrary, SPP can aggregate any image on any scale.

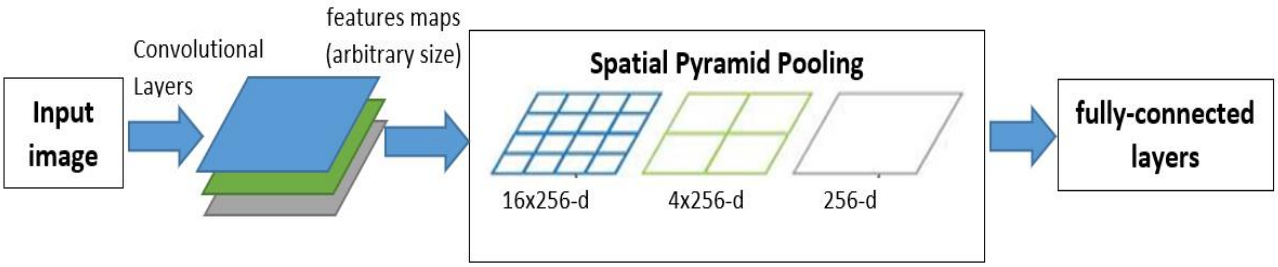


Fig. 1 SPP network layers.

The SPP block layer is added over the CSPDarknet53 in Yolo V4 and Yolo V4-tiny. The same SPP blocks are used for a spatial model configuration file. Further, the spatial model applies down sampling in the CNN layers to obtain the essential features in the max-pooling layers. It employs three distinct sizes of the max pool for each image by using [route]. Various layers -2, -4 and -1, -3, -5, -6 in $conv_5$ are applied in each [route] with the sizes 5×5, 9×9, and 13×13.

2.3 Path Aggregation Network (PAN)

At the beginning of deep learning, the model configuration is relatively easy and straightforward. Every layer receives input from the preceding layer. Next, the first layers obtain localized texture and pattern information to build up the last layer's semantic information. However, sometimes localized information that may be demanded to fine-tune the prediction may be lost. Consequently, the recent deep learning improvement, the interconnectivity between layers, is growing more complicated.

Path Aggregation Network (PAN) (S. Liu, Qi, Qin, Shi, & Jia, 2018) introduces a short-cut path which only takes about ten layers to go to the top N_5 layer (Tan, Pang, & Le, 2020). This shortcut makes fine-grain localized information accessible to top layers. Yolo V4 uses PAN as a parameter aggregation method from various backbone stages for different detector levels instead of the feature pyramid network (FPN) used in Yolo V3. Figure 2 displays the PAN network architecture. FPN introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P3 - P7). Hence, PANet adds an additional bottom-up pathway on top of FPN.

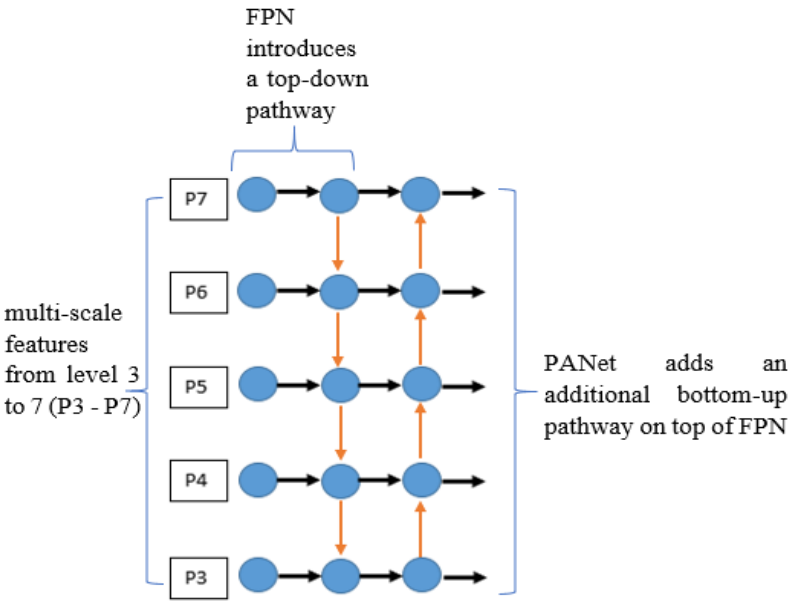


Fig. 2 PAN Network Architecture.

3. Methodology

3.1 Yolo V4 and Yolo V4-tiny

The latest version of Yolo is Yolo V4, released by (Bochkovskiy et al., 2020) in 2020. The Yolo V4 structure is: (1) Backbone: CSPDarknet53 (Wang, Liao, Wu, & Chen, 2020), (2) Neck: SPP (He, Zhang, Ren, & Sun, 2015) and PAN (S. Liu et al., 2018), and (3) Head: Yolo V3 (Redmon & Farhadi, 2018). In the backbone, Yolo V4 uses a Mish (Misra, 2019) activation function. Mish is a modern, smooth, and non-monotonic feature of the neural activation function, which can be seen in Equation (1) and Figure 3.

$$f(x) = x \tanh\left(\ln(1 + e^{x})\right) \tag{1}$$

Hence, $\ln(1 + e^{x})$ is the soft plus activation function (Q. Liu & Furber, 2016).

Bag of Freebies (BoF) is a technique that can make the object detector obtain more reliable accuracy without increasing the inference cost. These strategies only replace the training strategy or only update the training expense. Amendments can be made in the training method, including data augmentation, class imbalance, cost function, and soft labeling, to improve accuracy. These modifications do not affect the speed of inference.

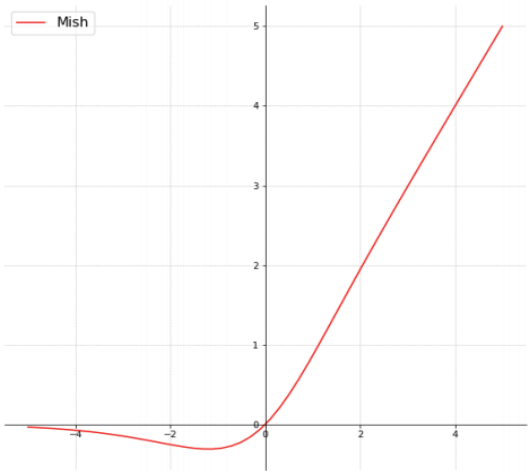


Fig. 3 Mish function.

By contrast, the Bag of Special (BoS) does not significantly affect the inference time with a strong output. These changes include enhancing the receptive field, the use of attention, the introduction of features such as skip connections & FPN, and postprocessing, such as non-maximum suppression (NMS). Complete Yolo V4 specifications are given in Table 1.

Yolo V4-tiny has been released in June 2020 (Alexey Bochkovskiy, 2020). We can use Yolo V4-tiny for a much faster training, detecting, and recognition processes. Furthermore, Yolo V4-tiny is especially useful if we have limited computing resources in either research or deployment. These models are also helpful if we are willing to trade off some detection and recognition performance for greater speed. The fundamental distinction between Yolo V4 and Yolo V4-tiny is that the network size is dramatically decreased. Moreover, the number of convolutional layers in the Cross Stage Partial Connections (CSP) backbone is compressed. Further, in our experiment the number of Yolo layers is two instead of three, and there are fewer anchor boxes for prediction. We employ two different scales of feature maps, 13×13 and 26×26, to predict the detection results.

Table 1. Yolo V4 specification.

Yolo V4	Backbone	Detector
Bag of Freebies (BoF)(Zhang et al., 2019)	• CutMix (Yun et al., 2019) and Mosaic data augmentation, • DropBlock (Ghiasi, Lin, & Le, 2018) regularization, • Class label smoothing.	• CIoU-loss, CmBN, • DropBlock regularization, • Mosaic data augmentation, • Self-Adversarial training, • Eliminate grid, • Sensitivity, • Using multiple anchors for a single ground/truth, • Cosine annealing scheduler (Loshchilov & Hutter, 2017), • Optimal hyperparameters, • Random training shapes.
Bag of Specials (BoS)	• Mish activation, • Cross-stage partial connections (CSP), • Multi input weighted residual connections (MiWRC).	• Mish activation, • SPP-block, • SAM-block, • PAN path-aggregation block, DIoU-NMS.

Yolo algorithm (Redmon & Farhadi, 2017) is a typical end-to-end network construction. This algorithm is shorter than the R-CNN algorithm [37, 38]. Yolo V4 adopts Yolo V3 as a one-stage dense prediction in the head. Further, Yolo V3 divides the input image into $S \times S$, where S consists of grids cells of the same size (Chen, He, Shi, & Zhong, 2019), and then predicts bounding boxes and possibilities for every grid cell. Further, Yolo V3 uses multiscale fusion to predict the image as a whole. This algorithm adopts a single CNN to prepare the entire image. The clusters are used to evaluate boundary lines.

Consequently, the K-means technique is selected to exhibit dimensional clusters within the target boxes' dataset and receive nine prior boxes of multiple sizes. Further, for all ground-truth objects, Yolo V3 permits the individual bounding box anchor. Thus, if the middle point of the object's base truth falls within a particular grid, the grid will identify the object.

Figure 4 demonstrates the bounding boxes with location prediction and dimension prior, as shown in Equations (2-5). b_x, b_y, b_w, and b_h are the x, y core coordinates, width, and height of our estimation. The network outputs are t_x, t_y, t_w, and t_k. Next, the grid's top-left coordinates are c_x and c_y, while p_w and p_h are the anchor dimensions for the box [15, 40, 41].

$$b_x = \sigma(t_x) + c_x \quad (2)$$

$$b_y = \sigma(t_y) + c_y \quad (3)$$

$$b_w = p_w e^{t_w} \quad (4)$$

$$b_h = p_h e^{t_h} \quad (5)$$

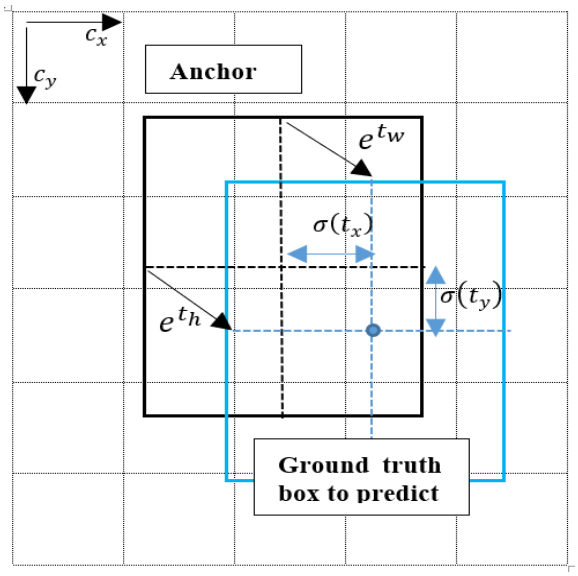


Fig. 4 Bounding boxes.

Yolo V3 predicts boxes at various scales using the theory of Feature Pyramid Networks (FPN). It employs a specific amount of CNN layers and residual layers to finish the detection and recognition process. Yolo V3 uses the features of the complete image to predict every bounding box. It also uses the K-means 9 cluster to calculate the anchor box [39, 42]. The purpose of K-means is to cluster the latitude to produce anchor boxes and to enable more significant *IoU* values in the adjacent ground truth, which is not related explicitly to the anchor box size. The K-mean clustering steps are shown in Algorithm 1.

Algorithm 1. The K-means clustering for anchor boxes.

1. Given K cluster center points: $(W_i, H_i), i \in \{1,2,3,4, \dots k\}$. Where (W_i, H_i) are the width and height of the anchor box.
2. Estimate the gap between ground truth and all cluster cored$(box, centroid) = 1 - IoU(box, centroid)$.
3. Recalculate the cluster center $W_i = \frac{1}{N_i} \sum W_i$, $H_i = \frac{1}{N_i} \sum H_i$. Further, N_i is the numbers of clusters.
4. Remade step 2 and step 3 until the groups gather.

3.2 Yolo V4 Architecture

In this step, we explain our proposed Taiwan Traffic Sign Dataset (TWTSD) recognition technique based on Yolo V4. Figure 5 demonstrates the Yolo V4 architecture. Taiwan Traffic Sign Dataset (TWTSD) class P1, P2, P3, and P4 were used as image inputs to the object detection method. Yolo V4 uses CSPDarknet53 as a backbone, SPP, and PAN as a neck, and Yolo V3 as a dense prediction in the head.

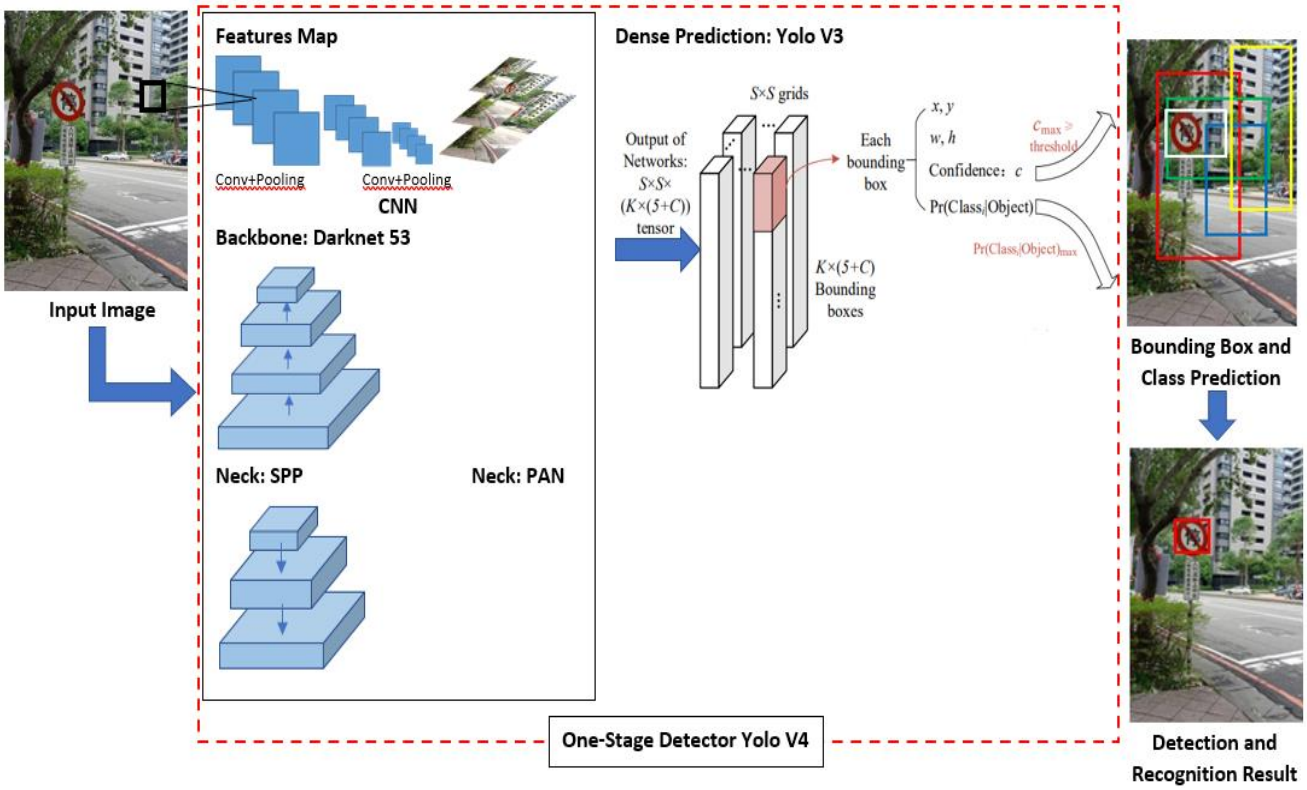


Fig. 5 One-Stage Detector Yolo V4.

The Yolo V4 architecture in Figure 5 processes several phases as follows:

(1) The first step is to separate the input image into (S×S) grids. Next, it builds K bounding boxes for each grid together with an estimate of the anchor boxes.

(2) CNN extracts all object features from the image. It then predicts the $\boldsymbol{b} = [\boldsymbol{b_x}, \boldsymbol{b_y}, \boldsymbol{b_w}, \boldsymbol{b_h}, \boldsymbol{b_c}]^{\boldsymbol{T}}$ and the $\boldsymbol{class} = [\boldsymbol{P1}, \boldsymbol{P2}, \boldsymbol{P3}, \boldsymbol{P4}]^{\boldsymbol{T}}$.

(3) Next, the system reviews the excellent confidence $\boldsymbol{IoU_{pred}^{truth}}$ of the *K* bounding boxes with the threshold $\boldsymbol{IoU_{thres}}$; If $\boldsymbol{IoU_{pred}^{truth}} > \boldsymbol{IoU_{thres}}$, the bounding box includes the object. Otherwise, the bounding box does not contain the object.

(4) Choose the category with the highest estimated probability as the target type.

(5) Non-Maximum Suppression (NMS) is used to perform a full local search to compress redundant boxes and output, then display the object detection results. To perform a full local search, the NMS is applied to overcome redundant boxes and output. Finally, the object identification results are determined.

Yolo V4 uses a spatial model to obtain the max-pooling layers' essential characteristics by sampling in convolutional layers. It uses three different maximum pool sizes for each image *[route]*. Further, various layers -2, -4 and -1, -3, -5, -6 in $\boldsymbol{conv_5}$ are used in each *[route]*.

3.3 Experiment Setup

Our experiment employed 4 CNN model Yolo V4 and Yolo V4-tiny to improve Taiwan traffic sign recognition. A detailed version of the experiment setup may be seen in Table 2. We delete the SPP layer in Yolo V4_2 and Yolo V4-tiny_2.

Table 2. Experiment setup.

No	Model	SPP
1	Yolo V4_1	Yes
2	Yolo V4_2	No
3	Yolo V4-tiny_1	Yes
4	Yolo V4-tiny_2	No

3.4. Taiwan Traffic Sign Dataset (TWTSD)

In this work, we use the Taiwan Traffic Sign Dataset (TWTSD) [9, 43] to evaluate our method. The traffic sign images were collected and processed manually from CarMax dashboard camera footage while driving on a sunny day and at night around Taichung City. The camera images, from which the traffic sign images are extracted, have a resolution of 1920×1080 pixels. The traffic sign images are cropped and annotated before use for training. The total number of images is 900, and their size varies. Statistics about the number of different traffic signs are given in Figure 6 (Yang, Long, Sangaiah, Zheng, & Tong, 2018).

The experiment employs a dataset that consists of 70% for training and 30% for testing. The dataset contains pictures of multiple scenes and signs. This experiment also focused on prohibitive signs from Taiwan, consisting of 235 no entry pictures, 250 no stop images, 185-speed limit pictures, and 230 no parking images. Figure 6 and Table 3 show the prohibitory signs of Taiwan in detail.

(a) (b) (c) (d)

Fig. 6 Taiwan's Prohibitory Signs (a) Class P1 (No Entry), (b) Class P2 (No Stopping), (c) Class P3 (No Parking), (d) Class P4 (Speed Limit).

Table 3. Taiwan's prohibitory signs

No	Class ID	Name	Training	Testing	Total
1	P1	No entry	165	70	235
2	P2	No stopping	175	75	250
3	P3	No parking	130	55	185
4	P4	Speed Limit	161	69	230

4. Experiment Result and Discussion

4.1 Training Results

Data augmentation is a vital aspect of training deep learning models with intensive research [45, 46]. The training procedure generated additional data from the primary images using simple geometrical transformations, including translations, rotations, scale changes, scoring, horizontal flips, and sometimes vertical flips. Those methods are commonly applied for large CNN training. Further,

the experiment conducts operations with several parameter settings during the data augmentation, such as rotation, zoom, width shift, and height shift. Yolo V4 uses CutMix, Mosaic data augmentation, and Class label smoothing as the Bag of Freebies backbone for classifier training.

This work enhances the model using a learning rate of 0.001 for analysis, a learning rate decay of 0.1 at each epoch, and 0.9 for momentum. Momentum is set to a value greater than 0.0 and less than one, where common values such as 0.9 and 0.99 are used in practice. Specifically, momentum values of 0.9 and 0.99 achieve reasonable train and test accuracy. Our experiment was conducted on a workstation equipped with an Nvidia RTX2080Ti GPU accelerator, 11 GB memory, i7 Central Processing Unit (CPU), and 16 GBDDR2 memory. The loss function must estimate the failure in the forecast and real value through the training process. The Yolo loss function of the predicted bounding box consists of four parts, and the formula is given below [47, 48, 49].

$$Loss = loss_1 + loss_2 + loss_3 + loss_4 \tag{6}$$

Further, $loss_1$ is the loss of predicted central coordinate and $loss_2$ is the loss of width and height of the prediction bounding box. Moreover, $loss_3$ is the loss of the predicted category, and $loss_4$ is the predicted confidence loss. The computation method for each part is given in Equations (7-10).

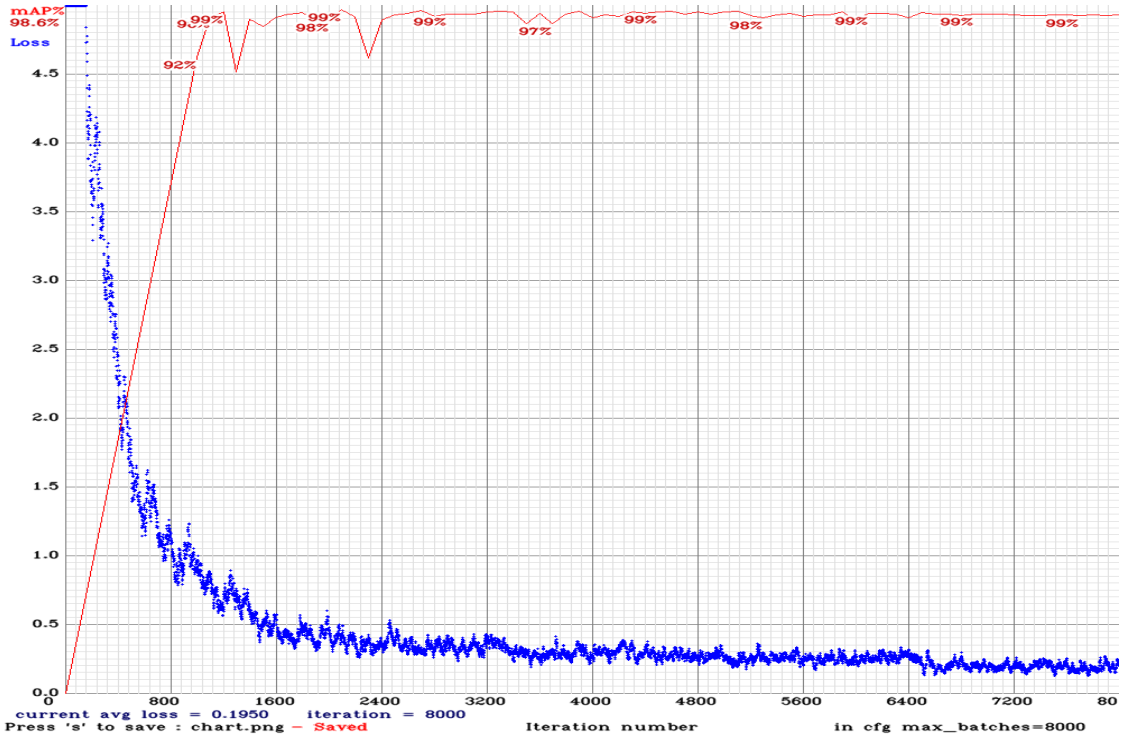
$$loss_1 = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{7}$$

$$loss_2 = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{8}$$

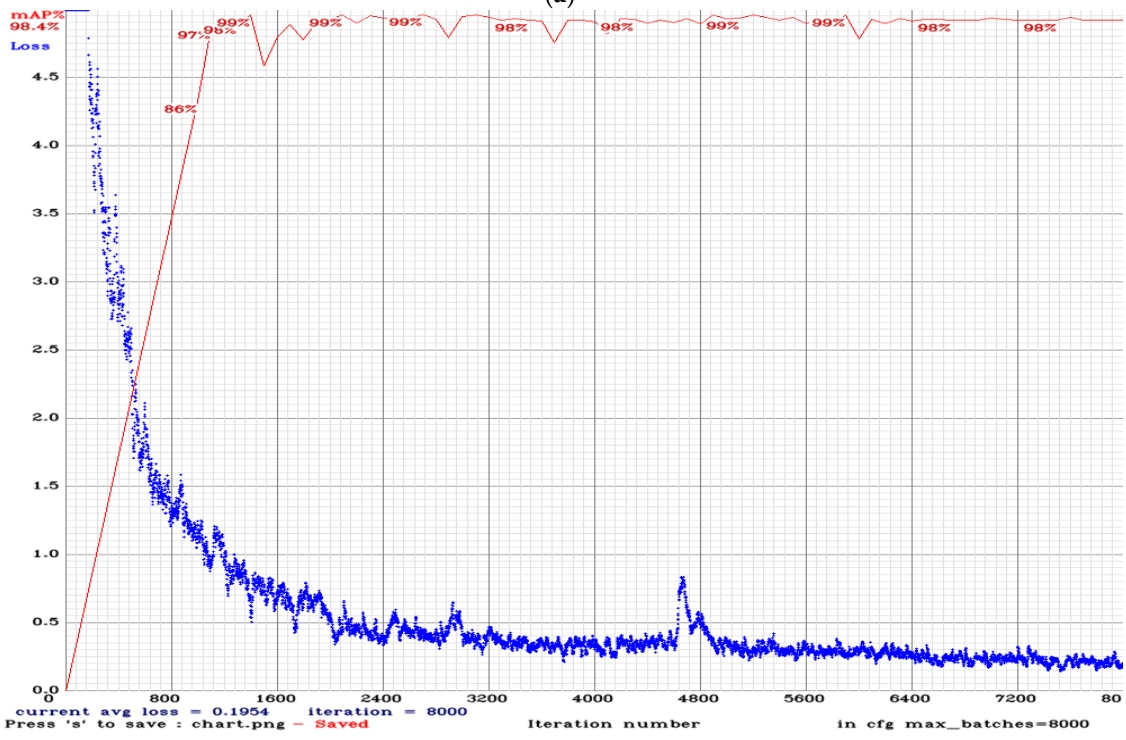
$$loss_3 = \sum_{i=0}^{s^2} I_{ij}^{obj} \sum_{c \epsilon class} (p_i(c), \hat{p}_i(c))^2 \tag{9}$$

$$loss_4 = \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} (c_i - \hat{c}_i)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} I_{ij}^{obj} (c_i - \hat{c}_i)^2 \tag{10}$$

Furthermore, (x_i, y_i) is the position of the prediction bounding box, s is the grid cell, B and j is the bounding box. The $(\hat{x}_i, \hat{y}_i)$ is the actual position obtained from the training data. w_i and h_i are the width and height of the predicted bounding box, respectively. λ_{coord} controls the prediction position loss of the prediction box. λ_{noobj} controls the no target loss in a single grid. c_i is the confidence score. $\hat{c}_i$ represents the intersection of the predicted bounding box and the actual box.

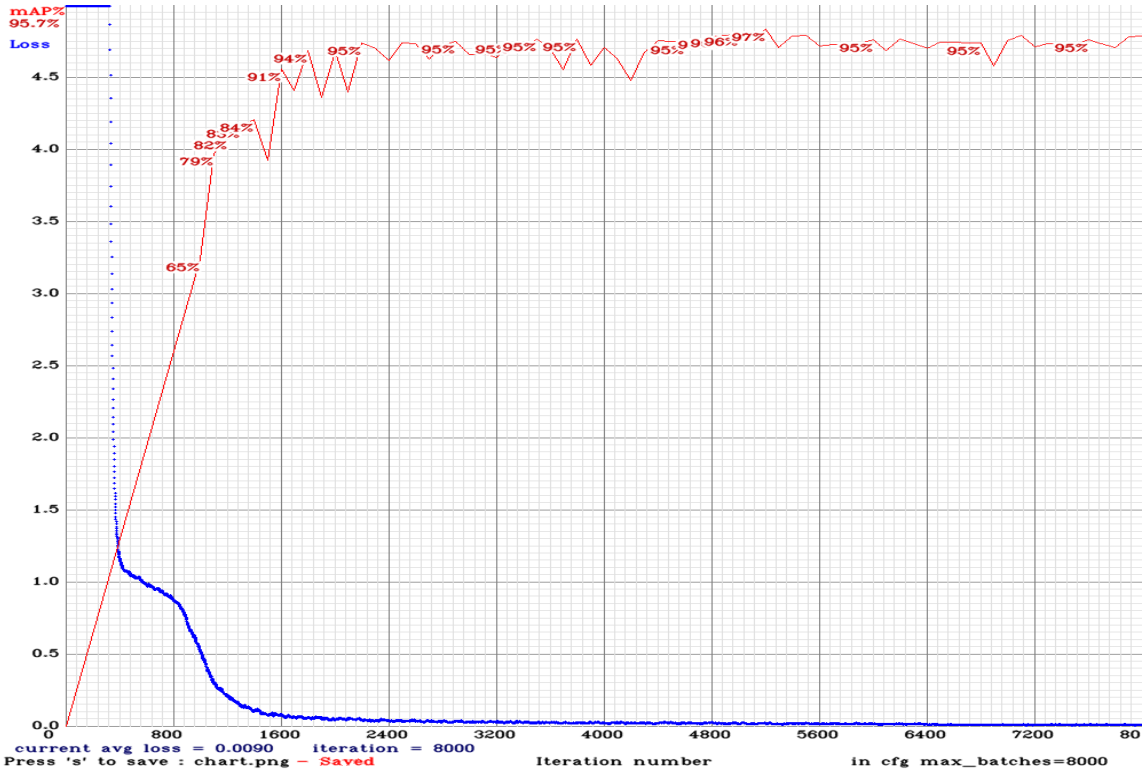


(**a**)

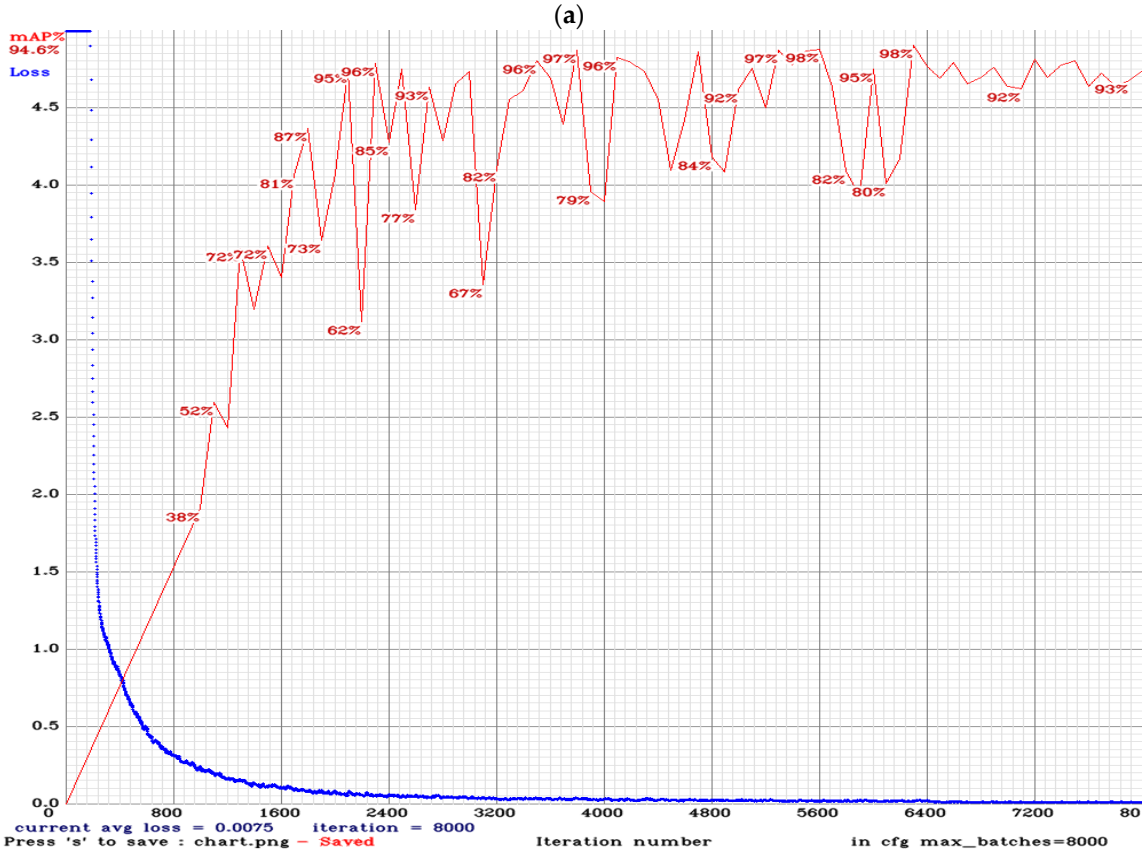


(b)

Fig. 7 Training performance evaluation by applying Yolo V4_1 (a) and Yolo V4_2(b).



(**a**)



(b)

Fig. 8 Training performance evaluation by applying Yolo V4-tiny_1(a) and Yolo V4-tiny_2(b).

Figure 7 presents the training process's reliability results for Yolo V4_1 (a) and Yolo V4_2 (b). Each model's training loss value is 0.1950 at 8000 iterations and 0.1954 at 8000 iterations, respectively. This experiment applies *learning_rate = 0.00261, max_batches = 8000 iterations, policy = steps, steps = 6400, 7200, scales = 0.1,0.1, momentum = 0.949, decay = 0.0005, and mosaic = 1*. The system starts with no information inside or zero information during the training process and uses a high learning rate value. As the neural network carries out its functions with rising data volumes, weights must be adjusted less aggressively.

The learning rate can be decayed to a small value close to zero. Alternately, the learning rate can be decayed over a fixed number of training epochs, then kept constant at a small value for the remaining training epochs to facilitate more time fine-tuning. Typically, m-class object detectors need 2000 × m as the maximum batches. In the experiment, the training process stops at 8000 iterations (2000 x 4 classes). Further, the scale (0.1, 0.1), and the current iteration number 10000 (0.001) batches are used in the training process. The calculation of the current learning rate becomes learning rate × scales [0] × scales [1] = 0.00001 and the learning rate value will be updated regularly.

The authenticity of the training process adopting Yolo V4-tiny_1 is presented in Figure 8a. The training loss value reaches 0.0090, and the training process terminates at 8000 iterations. During the training phase, Yolo V4-tiny_1 uses *learning_rate = 0.00261, burn_in = 1000, max_batches = 8000, policy = steps, steps = 6400,7200, scales = 0.1,0.1, momentum = 0.9, and decay = 0.0005*. In Figure 8b, Yolo V4-tiny_2 implements the same parameter settings as Yolo V4-tiny_1. During training, the iteration persists steadily at 4000 epochs and ends at 8000 iterations with the loss value at 0.0075.

Table 4. Training performance evaluation for all models.

Model	Loss Value	Class ID	AP (%)	TP	FP	Precision	Recall	F1-score	IoU (%)	mAP@0.50 (%)
Yolo V4_1	0.195	0	99.94	78	4	0.97	0.99	0.98	90.11	99.32
		1	100	83	0					
		2	98.39	62	0					
		3	98.96	76	4					
Yolo V4_2	0.1954	0	98.72	78	0	0.96	0.99	0.98	89	99.22
		1	100	83	2					
		2	98.39	62	1					
		3	99.76	76	9					
Yolo V4-tiny_1	0.009	0	95.67	62	0	0.98	0.91	0.94	75.69	98.63
		1	96.39	80	0					
		2	97.62	60	5					
		3	96.85	73	2					
Yolo V4-tiny_2	0.0075	0	93.3	73	2	0.97	0.97	0.97	72.92	97.08
		1	100	83	0					
		2	99.44	60	4					
		3	99.57	76	4					
Yolo V3 SPP (Dewi et al., 2020)	0.0125	0	97.5	78	0	0.99	0.99	0.99	90.09	98.88
		1	98.8	83	0					
		2	99.9	62	1					
		3	98.94	79	3					

Table 4 presents the training loss value, *mAP*, *AP*, precision, recall, F1, and *IoU* performance of the TWTSD Dataset. Yolo V4_1 obtains the highest *mAP*, around 99.32%, with an *IoU* of 90.11%, followed by Yolo V4_2 at 99.22% with an *IoU* of 89%, Yolo V4-tiny_1 at 98.63% with an *IoU* of

75.69%, and Yolo V4-tiny_2 at 97.08% with an *IoU* of 72.92%. Therefore, SPP can be combined and reinforced with any pattern. Yolo V3 SPP (Dewi et al., 2020) obtains 98.88 % accuracy for mAP with an *IoU* of 90.09. Thus, accuracy is 0.44% higher with Yolo V4_1.

Object detection metrics serve as a measure to assess how well the model performs on an object detection task. It also allows us to connect various detection systems objectively or compare them to a benchmark. In previous research, the average precision (AP) and its derivations are metrics selected to evaluate the detection and thus rank the teams. Precision (P) and Recall (R) are both very useful in recognizing what collection of documents or information is presented and how much of that document is useful for the question being asked. The precision (P) and recall (R) [50, 51] can be defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{11}$$

$$Recall = \frac{TP}{TP+FN} \tag{12}$$

where TP is an outcome in which the model correctly predicts the positive class. FP is an outcome in which the model incorrectly predicts the positive class, and FN is an outcome in which the model incorrectly predicts the negative class. The F1 [52, 53, 54] score was used to evaluate the performance of the model, and is shown in Equation (13).

$$F1 = \frac{2\ x\ Precision\ x\ Recall}{Precision+\ Recall} \tag{13}$$

The Average Precision (AP) and *mAP* formula is shown in Equation (14). The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

$$AP = \int_0^1 p(r)dr\ ,\ mAP = \frac{\sum AP}{NC} \tag{14}$$

where *p(r)* is the accuracy of target detection, NC indicates the target type, and N indicates the number of pictures *(C).* AP is calculated individually for each class. This means that there are as many AP values as the number of classes. Mean Average Precision (mAP) is the average of AP values over all classes.

The *Intersection over Union* (*IoU*) is an overlap metric that shows the overlap between the target created by the models and the original labeled framework [55, 56]. *IoU* is another criterion used to evaluate the detection accuracy. It can be obtained by calculating the overlap ratio between the prediction (*pred)* and ground-truth (*gt*) bounding boxes as follows [57, 46]:

$$IoU = \frac{Area_{pred} \cap Area_{gt}}{Area_{pred} \cup Area_{gt}} \tag{15}$$

4.2 Experiment Result

The precision and time measurements of the experiments are exhibited in Table 5 and Figure 9. Generally, Yolo V4_1 exhibits more dependable precision than other models. The SPP also affects the precision and detection time efficiency. The comprehensive test precision is represented by the TWTSD dataset, with different sizes, in Table 5 and Figure 9. Convolution and max-pooling have different advantages. Therefore, Convolution subsamples, perhaps in the corresponding sample layers, could be best reversed. Every [route] has been used to different layers -2, -4 and -1, -3, -5, -6 in. Nonetheless, Max Pooling works by only selecting maximum values from the neighboring regions to remove high-frequency noise from the image. Through their mix, both the benefits of establishing YoloV4 and Yolo V4-tiny backbone networks are influenced by the SPP. Table 5 and Figure 9 show testing accuracy using the TWTSD dataset. Yolo V4_1 reaches the highest average accuracy of

86.16%, followed by Yolo V4_2 at 80.09%, Yolo V4-tiny_1 at 74.41%, and Yolo V4-tiny at 72.59%. Class P2 exhibits a maximum average accuracy of 83.29%, followed by class P3 at 77.55%, class P1 at 76.53%, and class P4 at 75.88%.

Table 5. Testing accuracy on the TWTSD dataset.

Model	Testing Accuracy				
	P1	**P2**	**P3**	**P4**	**Average**
Yolo V4_1	0.9126	0.8446	0.8721	0.8170	0.8616
Yolo V4_2	0.8329	0.8103	0.7957	0.7646	0.8009
Yolo V4-tiny_1	0.6743	0.8441	0.7001	0.7579	0.7441
Yolo V4-tiny_2	0.6413	0.8326	0.7340	0.6956	0.7259
Average	0.7653	0.8329	0.7755	0.7588	

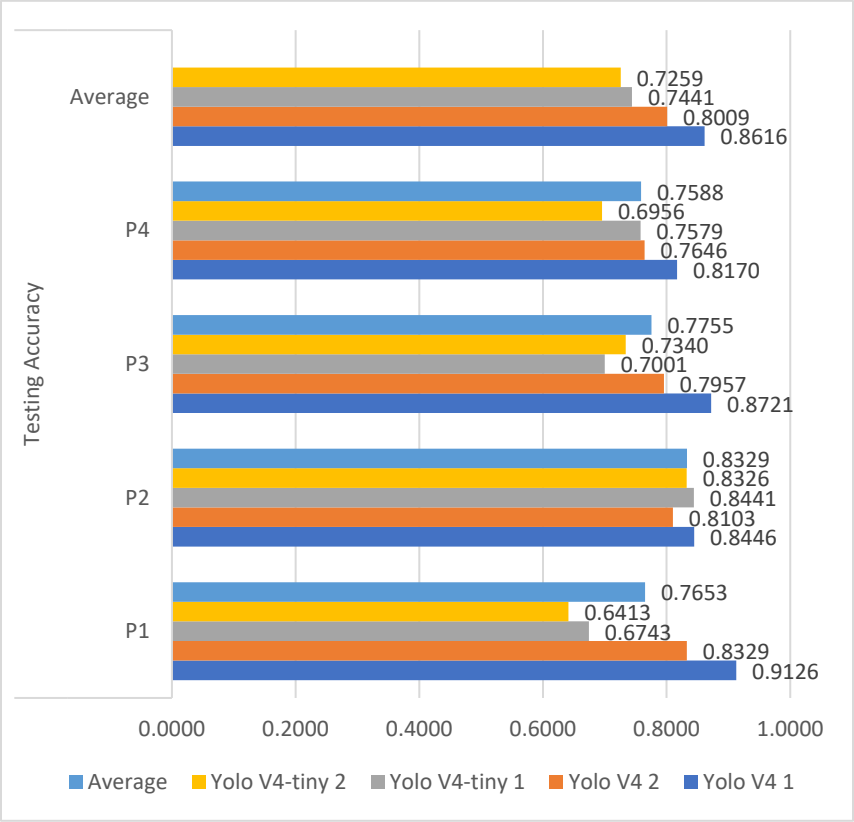


Fig. 9 Test set detection accuracy on the TWTSD dataset.

A comparison of our approach and previous research is given in Figure 10 and Table 6. In a previous study, the author (Dewi et al., 2020) used SPP with Yolo V3 to detect the TWTSD dataset and obtained 99% accuracy with 0.0176 seconds for the detecting time. Our proposed method outperforms other methods with 99.4% accuracy, and requires 0.0862 seconds for detecting the sign. Yolo V4_1 should need more time to detect the sign because it is loading more layers than the other methods (162 layers). The fastest model in the experiment is the Yolo V4-tiny_1. This model demands 0.0085 seconds to identify the sign and reaches 82.9% accuracy.

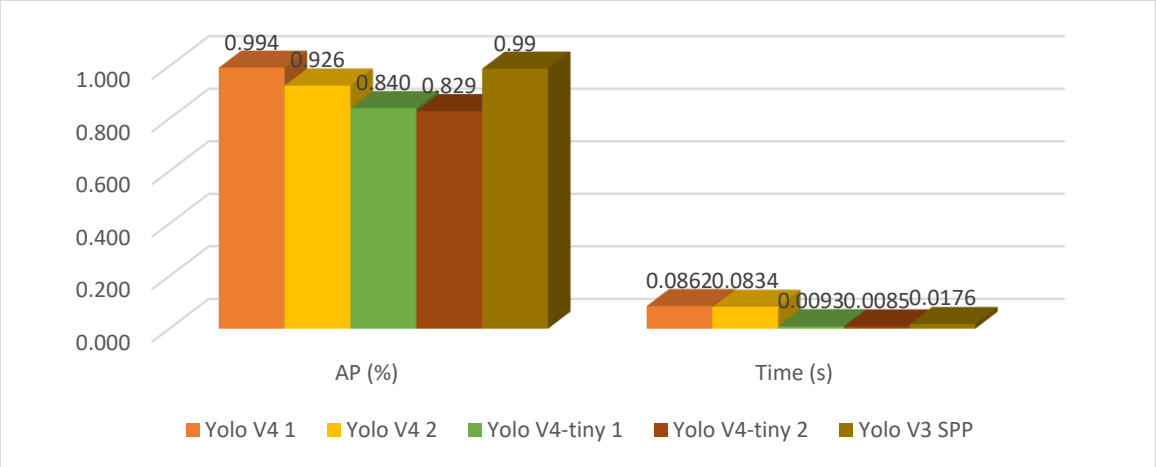


Fig. 10 The comparison of our approach and previous research in prohibitory sign detection.

Table. 6 Comparison of our approach and previous research in prohibitory sign detection.

Model	AP (%)	Time (s)
Yolo V4_1	0.994	0.0862
Yolo V4_2	0.926	0.0834
Yolo V4-tiny_1	0.840	0.0093
Yolo V4-tiny_2	0.829	0.0085
Yolo V3 SPP (Dewi et al., 2020)	0.99	0.0176

(a) **(b)** **(c)** **(d)**

Fig. 11 Recognition result Class P2 employing (a) Yolo V4_1, (b) Yolo V4_2, (c) Yolo V4-tiny_1, (d) Yolo V4-tiny_2.

Figure 11 presents the detection efficiency of the various algorithms in identifying Class P2. The localization precision of Yolo V4_1 in Figure 11a was greater than the others. Yolo V4_1 exhibits the highest accuracy, 99% with the position of the bounding box *left_x: 264, top_y: 4, width: 487,* and *height: 478*. It requires 90.637 milliseconds to detect the sign.

Fig. 12 Recognition result Class P4 employing (a) Yolo V4_1, (b) Yolo V4_2, (c) Yolo V4-tiny_1, (d) Yolo V4-tiny_2.

Figure 12 presents the Class P4 recognition results for several algorithms. Based on this result, all models can detect the sign correctly. Yolo V4_1 obtains the highest accuracy of 100% in Figure 12a, followed by Yolo V4_2, which exhibits 99% accuracy in Figure 12b. In Figure 12c, and Figure 12d, Yolo V4-tiny_1 and Yolo V4-tiny_2 respectively obtain 98% and 97% accuracy.

Fig. 13 Recognition result Class P3 employing (a) Yolo V4_1, (b) Yolo V4_2, (c) Yolo V4-tiny_1, (d) Yolo V4-tiny_2.

Figure 13a describes the Class P3 recognition results employing Yolo V4_1. The no entry sign was predicted in 258.462 milliseconds, with 98% accuracy. The bounding box coordinate position was *left_x: 8, top_y: 119, width: 84,* and *height: 67.* Applying an equivalent image, Yolo V4_2 in Figure 13b reaches 94% accuracy and needs 244.909 milliseconds for time detection. The coordinate position is *left_x: 4, top_y: 121, width: 85, and height: 63.* Moreover, detection results employing Yolo V4-tiny_1 and Yolo V4-tiny_2 are explained in Figure 13c, and Figure 13d, showing 97% and 92% accuracy, respectively. The outcome of the test in Figure 13 shows that both models can effectively distinguish class P3 with separate coordinate bounding boxes.

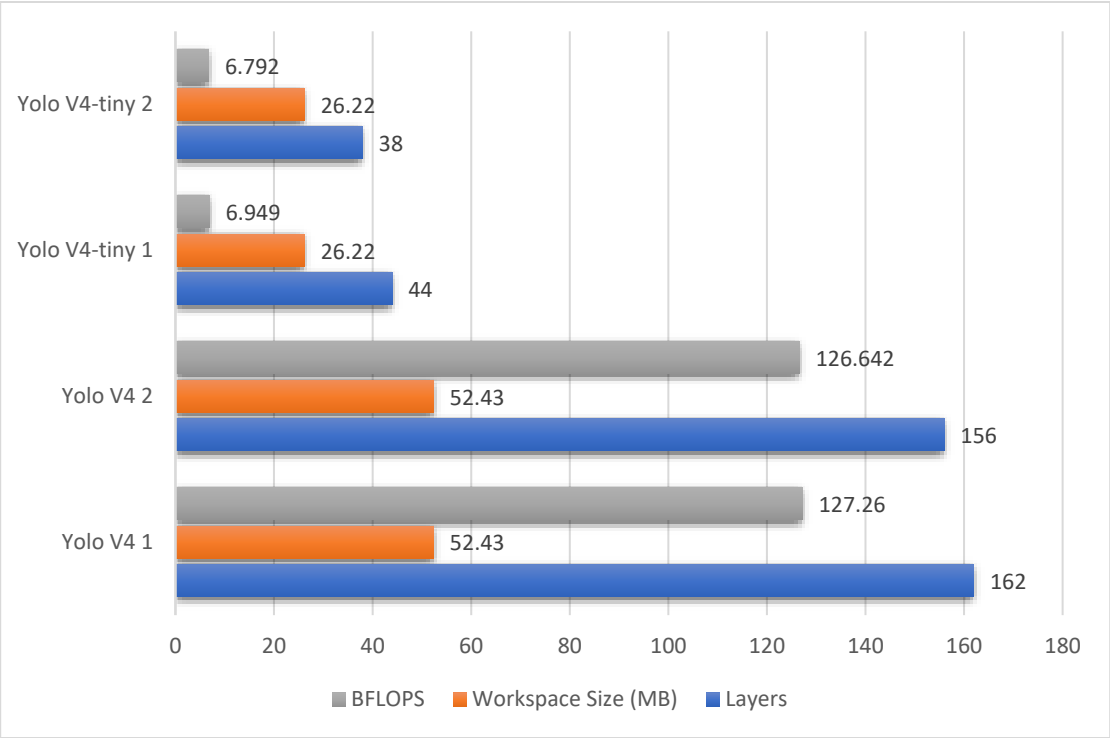


Fig. 14 The comparison of BFLOPS, workspace sizes, and layers.

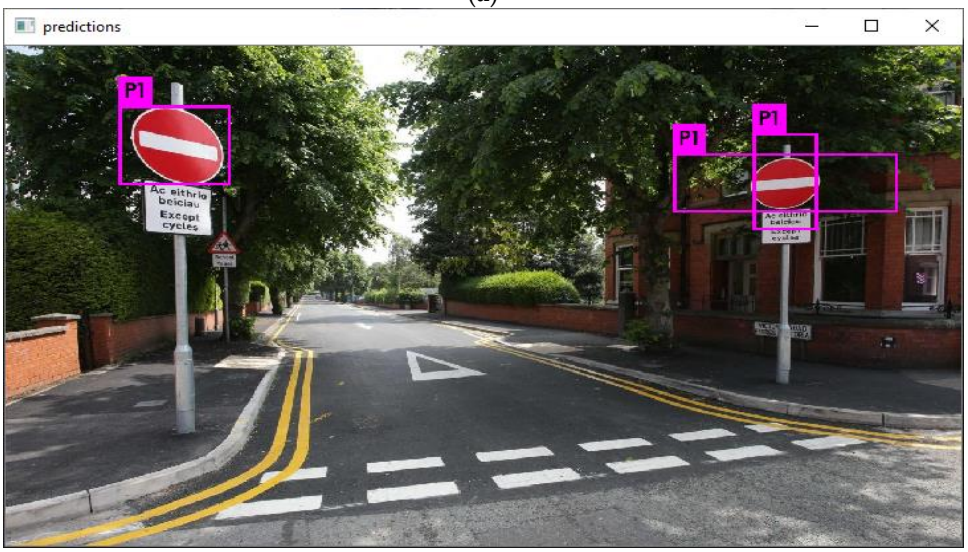
Figure 14 shows the evaluation of BFLOPS, workspace size, and layers for each experimental model. Yolo V4_1 has total BFLOPS of 127.26, allocated an extra workspace of 52.43 MB, and collected 162 layers from the weights-file. Yolo V4 fulfilled 156 layers and needs a workspace size of 52.43 MB with a total BFLOPS of 126.642. Furthermore, Yolo V4-tiny_1 and Yolo V4-tiny_2 loaded 44 layers and 38 layers, respectively, and present a small workspace size of 26.22 MB with a total BFLOPS of 6.949 and 6.792, respectively.

Consistent with previous research, Yolo V3 SPP (Dewi et al., 2020) load 114 layers, required a workspace of 52.43 MB, and a total BFLOPS of 65.69. Moreover, the best total BFLOPS reached by Yolo V4_1was 127.26, making Yolo V4_1 powerful, steady, and accurate. The results for recognition of Class P1 by Yolo V4_1 is shown in Figure 15a. Prediction took 252.145 milliseconds with 97% (*left_x: 154, top_y: 95, width: 112,* and *height: 113*) accuracy and 72% accuracy (*left_x: 924, top_y: 175, width: 121,* and *height: 68*). Additional workspace of 52.43 MB for Yolo V4_1 was needed, and 162 layers from the weights file were loaded, with a total BFLOPS of 90.245.

Figure 15b shows the recognition results using Yolo V4_2. Yolo V4_2 allocated additional workspace of 52.43 MB, loaded 156 layers from the weights file, obtained a total BFLOPS of 89.807, and required 238.499 milliseconds to detect the sign. This model draws two bounding boxes in one detection stage with the correct label Class P1 and obtained 95% accuracy (*left_x: 143, top_y: 90, width: 139,* and *height: 120*), 78% accuracy (*left_x: 838, top_y: 163, width: 282,* and *height: 87*), and 85% accuracy (*left_x: 938, top_y: 133, width: 82,* and *height: 145*). Figure 15c indicates the recognition result of Yolo V4-tiny_1. In Figure 15d, Yolo V4-tiny_2 failed to identify any class P1 signs in the image, recognizing only a single sign. Figure 15a shows that the location accuracy of Yolo V4_1 was superior to others.


predictions
P1
P1
Except cycles

(a)


predictions
P1
P1
P1
Except cycles

(b)


predictions
P1
P1
Except cycles

(c)

(d)

Fig. 15 Recognition result Class P1 employing (a) Yolo V4_1, (b) Yolo V4_2, (c) Tiny Yolo V4_1, (d) Tiny Yolo V4_2.

5. Conclusions

In this study, four models of traffic signs based on CNN were analyzed. Furthermore, this work analyzed and discussed all detectors' main contributions, including precision, detection time, workspace size, and total BFLOPS within CNN. Further, this research applied the Spatial Pyramid Pooling (SPP) and adjusted the Yolo V4 and Yolo V4-tiny backbone networks.

The experimental findings demonstrate that SPP can improve the efficacy of identifying the TWTSD prohibitory signs in Taiwan, and Yolo V4_1 is the most accurate model in the experiment. SPP boosts the YoloV4 and Yolo V4-tiny performance and backbone network. Our experiments demonstrate that Yolo V4_1 outperforms state-of-the-art schemes. Specifically, Yolo V4_1 obtained 99.4% accuracy in recognizing the Taiwan Traffic Sign Dataset (TWTSD). On average, Yolo V3 SPP is 0.4% higher than state-of-the-art Yolo V3 SPP (Dewi et al., 2020). The Yolo V3 SPP (Dewi et al., 2020) training process obtained 98.99 % accuracy for mAP with an *IoU* of 90.09. Accuracy thus rose by 0.44% with Yolo V4_1 (mAP 99.32%). While SPP takes more time, it is easier to detect multiple images in this model. As shown in Figure 15a, Yolo V4_1 can detect all signs in the image.

In future research, we will extend our dataset to all traffic signs in Taiwan. We will also combine traffic sign recognition with visual scene understanding based on deep explainable Artificial Intelligent (XAI) to generate better performance and results. We will use the *xCos* (Explainable Cosine) algorithm to provide an innovative verification for measuring similarity maps and discriminative position maps in future studies.

Acknowledgments

This paper is supported by the Ministry of Science and Technology, Taiwan. The Nos are MOST-107-2221-E-324 -018 -MY2 and MOST-109-2622-E-324 -004, Taiwan. This research is also partially sponsored by Chaoyang University of Technology (CYUT) and Higher Education Sprout Project, Ministry of Education (MOE), Taiwan, under the project name: "The R&D and the cultivation of talent for health-enhancement products."

Conflicts of Interest

The authors declared that they have no conflicts of interest in this work.

Reference

Alexey Bochkovskiy. (2020). Darknet: Open Source Neural Networks in Python. Retrieved January 13, 2020, from https://github.com/AlexeyAB/darknet

Avramović, A., Sluga, D., Tabernik, D., Skočaj, D., Stojnić, V., & Ilc, N. (2020). Neural-network-based traffic sign detection and recognition in high-definition images using region focusing and parallelization. *IEEE Access*, *8*. https://doi.org/10.1109/ACCESS.2020.3031191

Balali, V., Ashouri Rad, A., & Golparvar-Fard, M. (2015). Detection, classification, and mapping of U.S. traffic signs using google street view images for roadway inventory management. *Visualization in Engineering*, *3*(1), 1–18. https://doi.org/10.1186/s40327-015-0027-1

Bochkovskiy, A., Wang, C.-Y., & Mark Liao, H.-Y. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv:2004.10934*, 1–17. Retrieved from https://arxiv.org/abs/2004.10934

Chen, H., He, Z., Shi, B., & Zhong, T. (2019). Research on Recognition Method of Electrical Components Based on YOLO V3. *IEEE Access*, *7*, 157818–157829. https://doi.org/10.1109/ACCESS.2019.2950053

Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3642–3649. https://doi.org/10.1109/CVPR.2012.6248110

Dewi, C., Chen, R.-C., & Tai, S.-K. (2020). Evaluation of Robust Spatial Pyramid Pooling Based on Convolutional Neural Network for Traffic Sign Recognition System. *Electronics*, *9*(6), 889. https://doi.org/10.3390/electronics9060889

Ghiasi, G., Lin, T. Y., & Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. *Advances in Neural Information Processing Systems*, 10727–10737.

Guo, F., Qian, Y., & Shi, Y. (2021). Real-time railroad track components inspection based on the improved YOLOv4 framework. *Automation in Construction*, *125*. https://doi.org/10.1016/j.autcon.2021.103596

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

Liu, Q., & Furber, S. (2016). Noisy softplus: A biology inspired activation function. *Lecture Notes in Computer Science*, 405–412. https://doi.org/10.1007/978-3-319-46681-1_49

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8759–8768. https://doi.org/10.1109/CVPR.2018.00913

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Min, W., Li, X., Wang, Q., Zeng, Q., & Liao, Y. (2019). New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification. *IET Image Processing*, *13*(7), 1041–1049. https://doi.org/10.1049/iet-ipr.2018.6449

Misra, D. (2019). 1908.08681V2. *Mish: A Self Regularized Non-Monotonic Neural Activation Function*, (1).

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *CoRR*, *abs/1804.0*, 1–6. Retrieved from http://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Tai, S., Dewi, C., Chen, R., Liu, Y., Jiang, X., & Yu, H. (2020). Deep Learning for Traffic Sign Recognition Based on Spatial Pyramid Pooling with Scale Analysis. *Applied Sciences (Switzerland)*, *10*(19), 6997. https://doi.org/10.3390/app10196997

Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10778–10787.

Wang, C., Liao, H. M., Wu, Y., & Chen, P. (2020). CSPNet: A new backbone that can enhance learning capability of cnn. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop)*, 2.

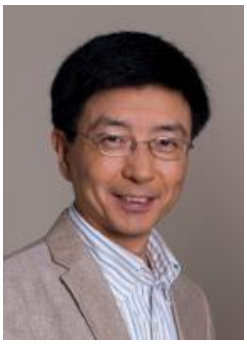
Yang, T., Long, X., Sangaiah, A. K., Zheng, Z., & Tong, C. (2018). Deep detection network for real-life traffic sign in vehicular networks. *Computer Networks*, *136*(8), 95–104. https://doi.org/10.1016/j.comnet.2018.02.026

Yun, S., Han, D., Chun, S., Oh, S. J., Choe, J., & Yoo, Y. (2019). CutMix: Regularization strategy to train strong classifiers with localizable features. *Proceedings of the IEEE International Conference on Computer Vision*, 6022–6031. https://doi.org/10.1109/ICCV.2019.00612

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., & Li, M. (2019). Bag of Freebies for Training Object Detection Neural Networks. *ArXiv:1902.04103v3*, 1–9.

Christine Dewi received a B.S. Degree (S. Kom.) from the Informatics engineering study program in 2010, and a Master of Computer Science (M.Cs.) from the Master of Information Systems study program in 2012, both from the Faculty of Information Technology, Satya Wacana Christian University, Salatiga, Indonesia. In 2018 she started her Ph.D. and is now Ph.D. Candidate at the College of Informatics, Chaoyang University of Technology, Taiwan. Her current research interests include Image Processing, Computer Vision, Object Detection and Recognition, Artificial Intelligence, and Machine Learning.

Rung-Ching Chen received a B.S. from the Department of Electrical Engineering in 1987, and an M. S. from the Institute of Computer Engineering in 1990, both from National Taiwan University of Science and Technology, Taipei, Taiwan. In 1998, he received his Ph.D. from the Department of Applied Mathematics in computer science, National Chung Hsing University. He is now a distinguished professor in the Department of Information Management, Taichung, Taiwan. His research interests include network technology, pattern recognition, and knowledge engineering, IoT and data analysis, and applications of Artificial Intelligence.



Xiaoyi Jiang received a bachelor's degree in computer science from Peking University, Beijing, China, and the Ph.D. and Venia Docendi (Habilitation) degrees in computer science from the University of Bern, Bern, Switzerland, in 1989 and 1997, respectively. He was an Associate Professor with the Technical University of Berlin, Berlin, Germany. Since 2002, he has been a Full Professor of Computer Science with the University of Münster, Münster, Germany, where he is currently the Dean with the Faculty of Mathematics and Computer Science. Dr. Jiang is a Fellow of the International Association for Pattern Recognition. He is currently the Editor-in-Chief for the International Journal of Pattern Recognition and Artificial Intelligence. Besides, he also serves on the Advisory Board and Editorial Board of several journals, including IEEE Transactions on Medical Imaging and International Journal of Neural Systems.

Hui Yu is a Professor at the CCI Faculty of the University of Portsmouth. His Ph.D. work won the Best Ph.D. Thesis Prize, EPSRC DHPA Awards and Vice-Chancellor Travel Prize, etc. He previously held a research appointment with the University of Glasgow. Moreover, his research interests include vision, computer graphics, and application of machine learning to these areas, particularly in research areas such as image/video processing and analysis, 3D/4D sensing, reconstruction and geometric processing, Human motion understanding, and effective analysis.