




Research Article

A Novel Neural Network Model for Traffic Sign Detection and Recognition under Extreme Conditions

Haifeng Wan ¹, Lei Gao ², Manman Su,¹ Qinglong You,³ Hui Qu,¹ and Qirun Sun ¹

¹School of Civil Engineering, Yantai University, Yantai, Shandong 264005, China

²CSIRO, Waite Campus, Urrbrae, SA 5064, Australia

³School of Highway, Chang'an University, Xi'an, Shaanxi 710064, China

Correspondence should be addressed to Haifeng Wan; wanhaifengytu2015@126.com

Received 1 April 2021; Revised 29 May 2021; Accepted 21 June 2021; Published 10 July 2021

Academic Editor: Manuel Aleixandre

Copyright © 2021 Haifeng Wan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Traffic sign detection is extremely important in autonomous driving and transportation safety systems. However, the accurate detection of traffic signs remains challenging, especially under extreme conditions. This paper proposes a novel model called Traffic Sign Yolo (TS-Yolo) based on the convolutional neural network to improve the detection and recognition accuracy of traffic signs, especially under low visibility and extremely restricted vision conditions. A copy-and-paste data augmentation method was used to build a large number of new samples based on existing traffic-sign datasets. Based on You Only Look Once (YoloV5), the mixed depth-wise convolution (MixConv) was employed to mix different kernel sizes in a single convolution operation, so that different patterns with various resolutions can be captured. Furthermore, the attentional feature fusion (AFF) module was integrated to fuse the features based on attention from same-layer to cross-layer scenarios, including short and long skip connections, and even performing the initial fusion with itself. The experimental results demonstrated that, using the YoloV5 dataset with augmentation, the precision was 71.92, which was increased by 34.56 compared with the data without augmentation, and the mean average precision mAP_{0.5} was 80.05, which was increased by 33.11 compared with the data without augmentation. When MixConv and AFF were applied to the TS-Yolo model, the precision was 74.53 and 2.61 higher than that with data augmentation only, and the value of mAP_{0.5} was 83.73 and 3.68 higher than that based on the YoloV5 dataset with augmentation only. Overall, the performance of the proposed method was competitive with the latest traffic sign detection approaches.

1. Introduction

Under severe weather conditions such as fog and snowy, traffic accidents occur frequently due to distracted driving, inattentiveness, or poor visibility [1]. To decrease the risk of accidents and improve the driving experience of drivers, traffic sign recognition systems (TSRS) have been developed and played an important role in autonomous driving and road network maintenance [2]. Traffic sign recognition systems can be categorized into two sub-tasks, i.e., detection and classification, the former aiming to identify the target objects from the images, and the latter aiming to classify the detected traffic signs into sub-classes [3].

An important step for the full automation of traffic sign recognition is using the automatic method to replace the manual method in localizing and recognizing traffic signs [4]. With the fast development of computer vision, especially the support of convolutional neural network (CNN), the detailed information from raw images can be extracted [5], and therefore, the traffic sign recognition has obtained great attention [4].

However, traffic sign recognition still faces the following challenges:

- (i) Under different weather conditions such as foggy, snowy, and rainy days, the captured traffic sign images may contain a lot of noises and distortions [6]

- (ii) Under different viewing angles and viewing distance, the traffic sign images may appear distortions of shape and color [6]
- (iii) The complicated road environment can lead to complex background of the traffic signs [1]
- (iv) Under different light conditions, color distortion or saturation may impact the quality of captured images [1]
- (v) Traffic signs may be partially hidden by trees, snow in some conditions

In order to address the above challenges, scholars have performed extensive studies. However, the traditional traffic sign recognition methods are not robust enough and have poor performance. For this reason, Girshick et al. firstly proposed the two-stage object recognition with significantly improved performance [7, 8]. In this method, some generic object proposals were calculated first, and then these candidates were classified [9]. The typical models also include Fast R-CNN [10], Faster R-CNN [11], and R-FCN [12].

One-stage detector was proposed to improve calculation efficiency. The one-stage detector does not have a regional proposal stage; instead, it predicts the object's class and position and obtains the final result through single stage. The most representative models are YOLO [13–18], SSD [19], and RetinaNet [20]. The one-stage traffic sign recognition model has high efficiency but insufficient accuracy [1].

The purpose of this study is to develop a high recognition accuracy method for traffic recognition system which can handle special situations such as severe weather and bad light conditions. In this paper, a new traffic sign recognition method based on YoloV5 was proposed. The main contributions can be summarized as follows:

- (i) The detection and recognition of traffic signs under extreme conditions is one of the technical bottlenecks of automatic driving and intelligent transportation. Through experimental research and test, this paper provided a scientific means and framework for accurate recognition of traffic signs, which can significantly increase the driving safety, especially under extreme conditions.
- (ii) About 1000 images under severe weather conditions in Shandong were captured and annotated, greatly enriching the dataset. The copy and paste data augmentation was used to construct a large number of new samples based on existing traffic-sign instances, which allows the training and learning for small dataset and also improves the variability of the traffic sign data samples. Meanwhile, images with different weather conditions were captured and augmented as well, solving a lot of problems such as noises and distortions in the images.
- (iii) Based on YoloV5, the TS-Yolo model was proposed. MixConv [21, 22] was used which could do the convolutional operation with mixed kernel sizes, so

that different patterns with various resolutions can be easily captured. Thus, in different viewing angles and distances, even if the traffic signs appear distorted in shapes and colors, they still could be detected

AFF [23] module was also used to fuse features based on attention which came from same-layer or cross-layer, including short and long skip connections, and even perform the initial fusion with itself. Therefore, under complex road conditions or different light conditions, even sometimes the traffic signs were partially hidden, and using the enhanced model, images can be recognized and localized more accurately.

- (iv) Experimental results on public datasets demonstrated that the proposed approach had comparable accuracy with the latest traffic sign detection methods. Therefore, the proposed method can be applied in practical use for driving assistance

The rest of the paper is organized as follows: Section 2 reviews the related work about traditional traffic sign recognition technology and deep neural networks; Section 3 describes the proposed method and TS-Yolo model in detail; Section 4 describes dataset and experiment setup; Section 5 presents the experiment results and the analysis results; Section 6 summarizes the conclusions and our findings; Section 7 looks at the future research prospects and suggestions.

2. Related Work

2.1. Traditional Traffic Sign Recognition Technology. The traditional recognition algorithms are aimed at locating the region of interest and identifying the classification [1]. In the study by Zhou and Deng [24], color and spaces were combined so that the traffic sign colors can be treated as one class. Li et al. [25] developed a new traffic sign detection method by integrating the image segmentation based on color invariants and the shape matching based on pyramid histogram of oriented gradients (PHOG) features. Based on support vector machines (SVM), Maldonado-Bascón et al. [26] proposed an automatic road-sign detection and recognition system. Salti et al. [5] addressed the problem of traffic sign detection in mobile mapping data by combining solid image analysis and pattern recognition techniques. The proposed method was based on the extraction of interest regions instead of sliding window detection. Lillo-Castellano et al. [27] proposed a three-stage system, which included segmentation of chromatic and achromatic scene elements using $L * a * b *$ and HSI spaces, discarding noninterest regions in postprocessing, and sign-shape classification using Fourier descriptors.

2.2. Traffic Sign Recognition Based on Deep Neural Networks. In recent years, deep neural networks (DNNs) have received great attention in pattern recognition and computer vision research [28] and have been widely used in both object detection and recognition [29–35].

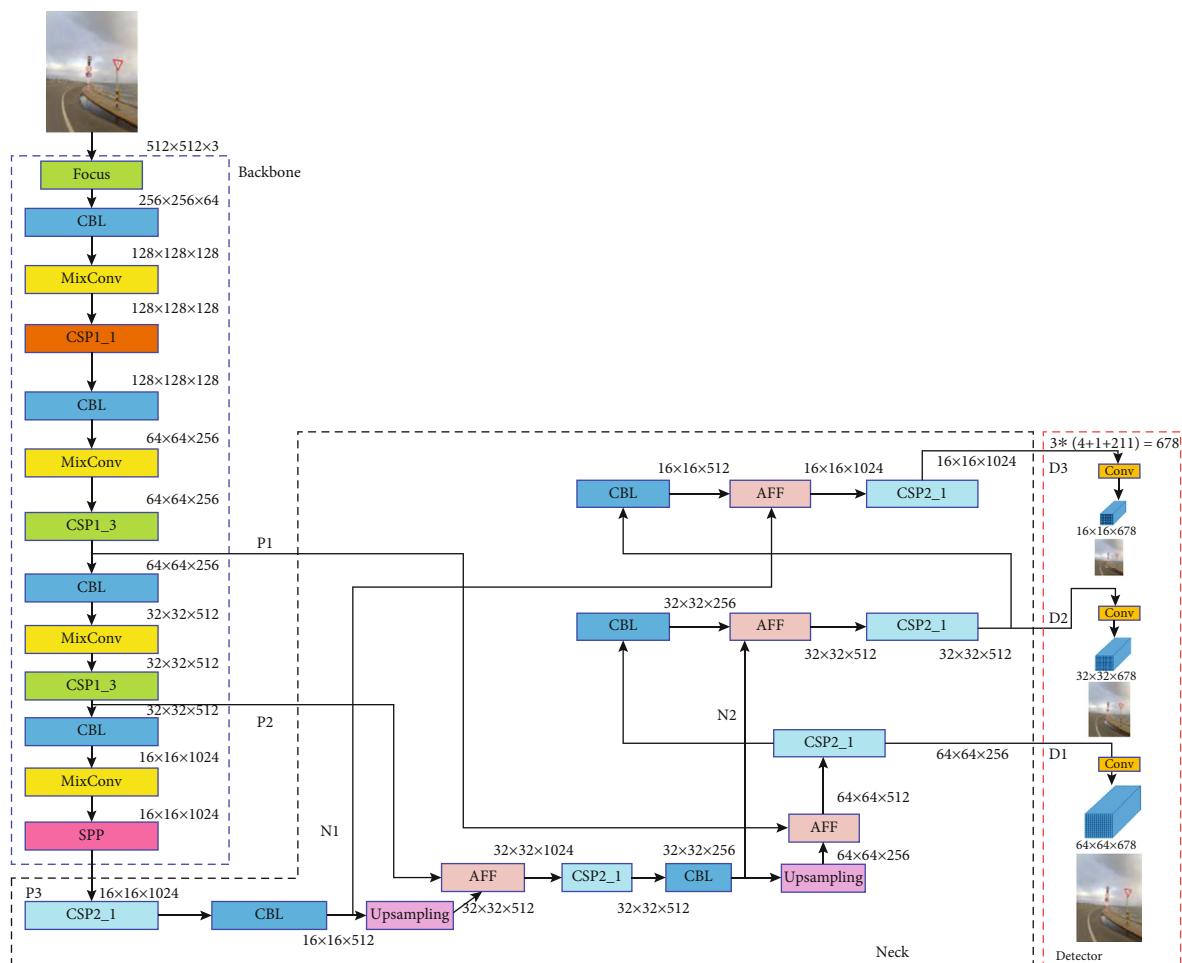


FIGURE 1: Overall architecture of the proposed approach.

Belghaouti et al. [36] proposed an automatic road sign recognition system based on LeNet model and got 99% accuracy in German traffic dataset. Yang et al. [37] developed a novel end-to-end deep network, which used a two-stage adjusting strategy to extract region proposals. Song et al. [38] proposed an efficient convolutional neural network (CNN), which can significantly reduce the redundancy, reduce the parameters, and improve the speed of the networks. In the study by Alghmgham et al. [39], automatic Arabic traffic sign (AATS) recognition system was designed using convolutional neural networks (CNN). Arcos-García et al. [40] presented a two-stage traffic sign recognition system with high efficiency. In the system, a LINX Mobile Mapper system was firstly used to acquire and process 3D point cloud data. Then, a deep neural network was used to classify the point cloud projection on RGB images. Zhou et al. [41] proposed an ice environment traffic sign recognition benchmark (ITSRB) and detection benchmark (ITSDB) marked in the COCO2017 format. In addition, a high-resolution traffic sign classification (PFANet) based on attention network was proposed, and ablation research was performed on the design parallel fusion attention module. Pei et al. [42] proposed Multiscale Deconvolution Networks (MDN), which can integrate multiscale convolutional neural

network and deconvolution subnetwork. On that basis, a localized traffic sign recognition model training with high efficiency and reliability can be obtained. Zhu et al. [43] proposed a novel framework for object classification, which contained two deep learning components, i.e., fully convolutional network (FCN) and deep convolutional neural network (CNN). Chaudhari et al. [44] put forward an approach to recognize traffic signs using small-scale deep convolutional neural networks (CNN), which can be applied to different applications. Franzen et al. [45] used neural networks for traffic sign recognition. Compared with existing works, our method is unique in that the traffic sign recognition is performed in the frequency domain instead of the spatial domain.

The above approaches have achieved high accuracy for traffic sign recognition based on different datasets. However, the recognition rates largely depend on the application context, regardless of the severe weather conditions.

3. Proposed Method

3.1. Overall Architecture. The overall architecture is shown in Figure 1. As the figure shows, the system is composed of backbone, neck, and detector. In order to detect the position and class of an object, it is necessary to extract features from

the image and capture the features using backbone for positioning and classification. MixConv [22] was used which could do the convolutional operation with mixed kernel sizes, so that different patterns with various resolutions can be captured.

With the initial output features of backbone, neck fuses features and adapts size, thus, improving the overall performance of the architecture. AFF [23] was used in neck layer to fuse the features which came from same-layer or cross-layer based on attention, including short and long skip connections, and even performs the initial fusion inside itself. The detector receives the three outputs from neck network and outputs the prediction of bounding box position, object confidence, and object classes for each feature map output layer. With the help of high-quality feature maps of each layer, the detection can be more accurate.

3.2. Backbone Network. The backbone was composed of Focus, Convolution with Batch normalization and LeakyRelu (CBL), Mix Convolution (MixConv), Cross Stage Partial Network (CSP), and Spatial Pyramid Pooling (SPP). The size of the input image was $512 \times 512 \times 3$, among which 512×512 represented the width and height in pixels, respectively, and 3 represented 3 channels. After the focus module, the size was changed to $256 \times 256 \times 64$. After the following CBL module, the size became $128 \times 128 \times 128$.

The following MixConv module and CSP1_1 module did not change the size, and the image size was still $128 \times 128 \times 128$ after these two modules.

The next step was a CBL module, which changed the image size to $64 \times 64 \times 256$. Then, the Mix Convolution module did not make any change to the image, and the size was still $64 \times 64 \times 256$. After that, the CSP1_3 module did not change the output size. Thus, the output with the size of $64 \times 64 \times 256$ was recorded as P1.

The next step was a CBL module, which changed the output size to $32 \times 32 \times 512$. Then, a MixConv module did not make any changes to the size, and the CSP1_3 module in the following step did not change the size either. The output here with the size of $32 \times 32 \times 512$ was recorded as P2.

The next step was a CBL module, which changed the size to $16 \times 16 \times 1024$. Then, the MixConv module and the SPP module in the following step did not change the output size. The output here with the size of $16 \times 16 \times 1024$ was recorded as P3.

3.3. Neck Network. The neck was composed of CBL, CSP, Upsampling, and AFF. The outputs of backbone, namely, P1, P2, and P3, were used as the input.

First, the output P3 with the size of $16 \times 16 \times 1024$ was input into the CSP2_1 module, and the image size was still $16 \times 16 \times 1024$. Then, the CBL module in the next step changed the size to $16 \times 16 \times 512$. The output here was recorded as N1.

Then N1 was input into the Upsampling module, and the size was changed to $32 \times 32 \times 512$. Both output from Upsampling and the output P2 from backbone had the same size of $32 \times 32 \times 512$ and were used as the inputs to the AFF module. The output of AFF had a size of $32 \times 32 \times 1024$. Then, the

output was input into the CSP2_1 module, and the size was changed to $32 \times 32 \times 512$. A CBL module in the next step changed the size to $32 \times 32 \times 256$. The output here was recorded as N2.

The next step was an Upsampling module, which changed the output size to $64 \times 64 \times 256$. The output from Upsampling and the output P1 from the backbone network both had the size of $64 \times 64 \times 256$ and were used as the inputs to the AFF module. The output of the AFF module had a size of $64 \times 64 \times 512$. Then, the CSP2_1 in the next step changed the size to $64 \times 64 \times 256$. The output here was recorded as D1.

The next step was a CBL module, which changed the output size to $32 \times 32 \times 256$. The output from the CBL module and the previous output N2 both had the size of $32 \times 32 \times 256$ and were used as the inputs to the AFF module. The output of the AFF module had a size of $32 \times 32 \times 512$. The CSP2_1 in the next step did not make any change to the size. The output here with the size of $32 \times 32 \times 512$ was recorded as D2.

The next step was a CBL module, which changed the size to $16 \times 16 \times 512$. The output of CBL and the previous output N1 both had the size $16 \times 16 \times 512$, so they were used as two inputs to the AFF module. The output of the AFF module had a size of $16 \times 16 \times 1024$. The CSP2_1 module in the next step did not make any change to the size. The output here with the size of $16 \times 16 \times 1024$ was recorded as D3.

3.4. Detector. The detector received the three outputs from neck network, which were D1 ($64 \times 64 \times 256$), D2 ($32 \times 32 \times 512$), and D3 ($16 \times 16 \times 1024$). Each output was input into a CBL unit to adjust the output channels; then, the prediction box position, confidence, and object classes of every grid point for each feature map output layer were obtained. The position of the top left point of the predicted box was denoted as $[x_{\min}, y_{\min}]$, and the position of the bottom right point was denoted as $[x_{\max}, y_{\max}]$. Confidence refers to whether the bounding box has the target object; object classes refer to the probability that the target belongs to each class inside the grid box. In this case, the object classes can be calculated as 4 (position points) + 1 (confidence) + 221 (traffic sign classes + background) = 226 . There were 3 anchor boxes for each position, so the output channel was 678.

3.5. Components of Backbone and Neck. The focus module is depicted in Figure 2. The focus module took the input image with the size of $512 \times 512 \times 3$ and performed slicing operation. This operation retrieved every other pixel from an image, similar to downsampling. Then, 4 images were obtained from an input image, the 4 images were complement with each other, and thus, there was no data missing. After that, the width information and height information were concatenated into the channel space, and the input space was expanded 4 times. Therefore, the concatenated image had 12 channels while the original image had only 3 channels. Finally, the new image was convolved to produce two downsampling feature maps without any data missing. In this case, a feature map with the size of $256 \times 256 \times 12$

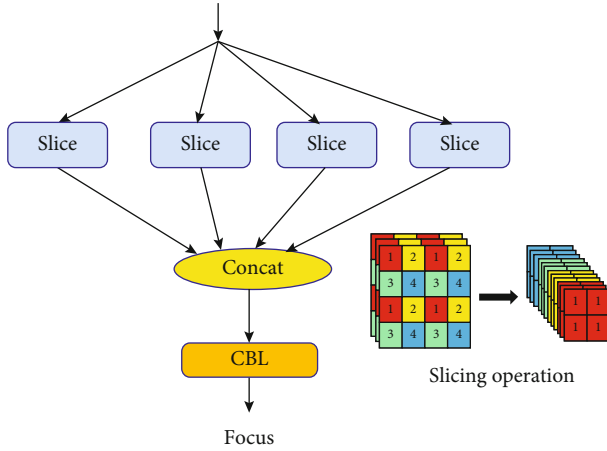


FIGURE 2: Focus module.

was obtained. As shown in the bottom right in Figure 2, after image slicing operation, the image with the size of $4 \times 4 \times 3$ was changed to a feature map with the size of $2 \times 2 \times 12$. After convolution with the channel of 64, the output with the size of $256 \times 256 \times 64$ was obtained. Focus module can help reduce the computational workload of downsampling, but it will not introduce any data missing. Therefore, the focus module can retain more complete picture downsampling data for the following feature extraction.

As shown in Figure 3(a), CBL in backbone was composed of a convolution function with Batch normalization and LeakyRelu. Residual unit (ResUnit) was used in CSP. As shown in Figure 3(b), ResUnit was composed of two CBL units, which were connected continuously. The original input and the output of the 2nd CBL performed the vector addition function as the output.

CSP1_X is depicted in Figure 3(c). The original input of CSP1 went through a CBL module and X modules of ResUnit. It finally performed a convolutional function to obtain the temporary output of the main path. Meanwhile, the original input performed another convolutional function, went through another path, and then concatenated with the output of the main path. The result went through batch normalization, LeakyRelu, and CBL.

CSP2_X is described in Figure 3(d), and the structure is slightly different from CSP1_X. In CSP2_X, the main path consisted of $2 * X$ times CBL units instead of ResUnit in CSP1_X. The original input of CSP2_X went through $2 * X$ CBL units and then a convolutional function, to obtain the temporary output of the main path. Meanwhile, the original input performed another convolutional function, went through another path, and then concatenated with the output of the main path. The result went through Batch normalization, LeakyRelu, and CBL.

In summary, the main idea of Cross Stage Partial Network (CSP) is to produce two paths for the input. The main path has CBL or ResUnit, another path will perform convolutional function, and the results from the two paths will be merged. This strategy can reduce the computational costs. In detail, the CSP improves the learning capability of the convolutional neural network, reduces the computational

complexity, ensures the high accuracy and light weight, and reduces the memory cost. CSP can integrate the gradient changes into the feature map from the beginning to end, which can reduce the computational costs while ensuring the accuracy.

Spatial Pyramid Pooling (SPP) [46] is used to realize the integration of local features and global features, thereby improving the representation ability of the feature map. SPP module is depicted in Figure 4. The input went through a CBL and then through three separate paths. The three max pooling functions had kernel sizes of 5, 9, and 13. The three outputs will be merged and then obtain the image with the same size as the input. Finally, the obtained merged output was input to a CBL to obtain the final output with the same size as the input.

MixConv [22] mixed different kernel sizes (3×3 , 5×5 , and 7×7) in a single convolution operation, so that different patterns with various resolutions can be easily captured. MixConv is described in Figure 5. The input tensor has the shape of (h, w, c) , and it is denoted as $X^{(h,w,c)}$, where h is the height, w is the width, and c is the channel size. A depth-wise convolutional kernel can be denoted as $W^{(k,k,c,m)}$, where $k \times k$ refers to the kernel size, c refers to the input channel size, and m refers to the channel multiplier.

In order to simplify the analysis, in this study, we assumed that the kernel had the same width and height (k). With this assumption, the same shape (h, w) and multiplied output channel size $m \cdot c$ were obtained for the output tensor $Y^{(h,w,c \cdot m)}$. The feature map of each output can be calculated by the following equation (1) [22]:

$$\hat{Y}_{x,y,z}^t = \sum_{-k_i/2 \leq i \leq k_i/2, -k_i/2 \leq j \leq k_i/2} \hat{X}_{x+i,y+j,z/m}^t \cdot \hat{W}_{i,j,z}^t, \forall z = 1, \dots, m \cdot c_t. \quad (1)$$

As shown in Figure 5, the channels were divided into several groups by MixConv, and different kernel sizes were applied to different groups. More specifically, the input tensor was divided into g groups of virtual tensors, i.e., $\langle X \wedge^{(h,w,c_1)}, \dots, X \wedge^{(h,w,c_g)} \rangle$. The spatial height and width of all virtual tensors were the same, which were h and w , respectively. In addition, the total channel size of these virtual tensors was the same as the original input tensor, i.e., $c_1 + c_2 + \dots + c_g = c$. Similarly, the convolutional kernel was also partitioned into g groups of virtual kernels $\langle W \wedge^{(k_1,k_1,c_1,m)}, \dots, W \wedge^{(k_g,k_g,c_g,m)} \rangle$.

The corresponding virtual output for the t -th group of virtual input tensors and kernels can be calculated by equation (2):

$$\hat{Y}_{x,y,z}^t = \sum_{-k_i/2 \leq i \leq k_i/2, -k_i/2 \leq j \leq k_i/2} \hat{X}_{x+i,y+j,z/m}^t \cdot \hat{W}_{i,j,z}^t, \forall z = 1, \dots, m \cdot c_t. \quad (2)$$

The final output tensor can be obtained by concatenating all virtual output tensors:

$$Y_{x,y,z_o} = \text{Concat}(\hat{Y}_{x,y,z_1}^1, \dots, \hat{Y}_{x,y,z_g}^g). \quad (3)$$

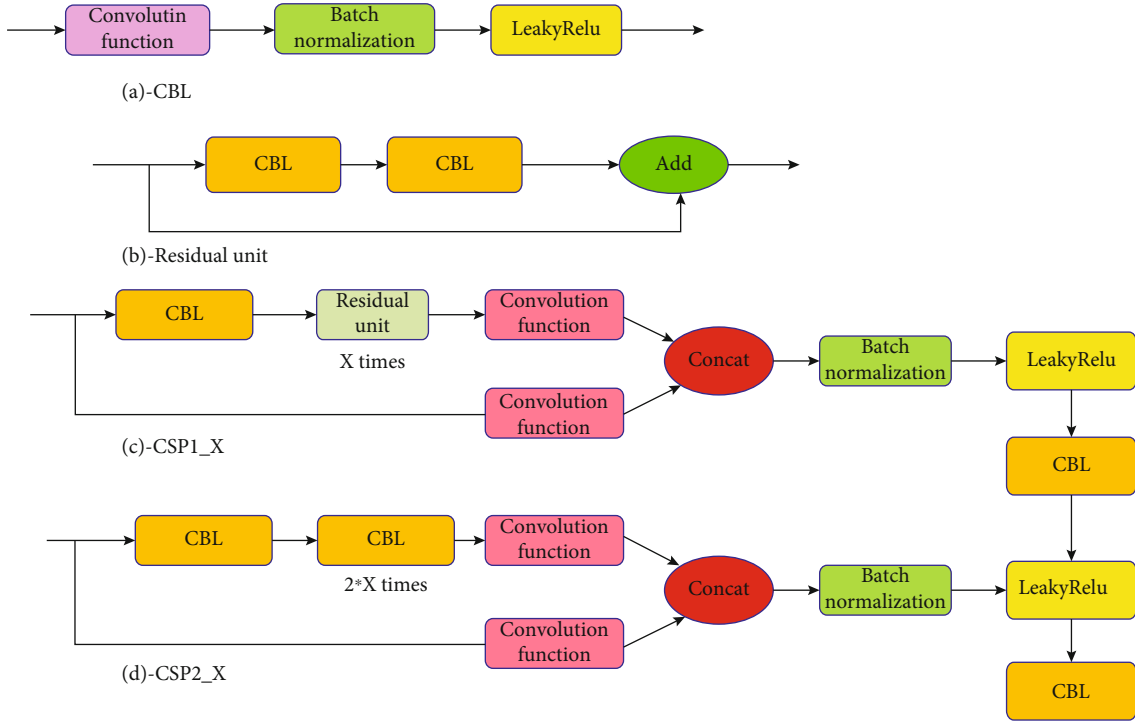


FIGURE 3: (a) CBL. (b) Residual Unit. (c) CSP1_X. (d) CSP2_X.

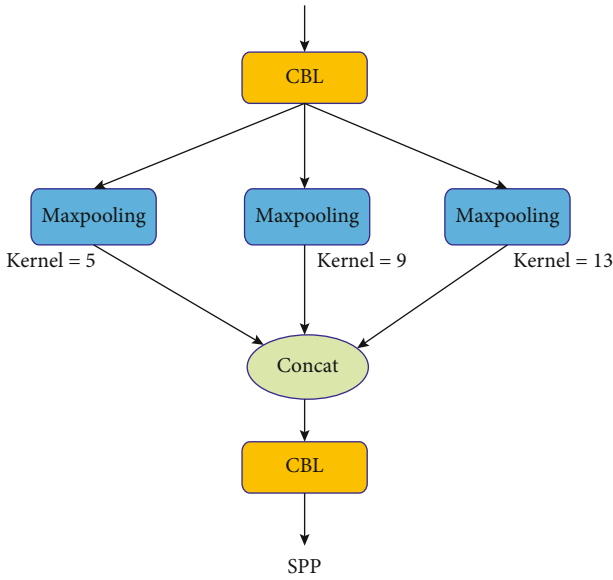


FIGURE 4: SPP module.

The channel size of the final output is $z_o = z_1 + \dots + z_g = m \cdot c$.

MixConv mixed different kernel sizes (3×3 , 5×5 , and 7×7) in a single convolution operation, so that different patterns with various resolutions could be easily captured.

As shown in Figure 6, multiscale channel attention module (MS-CAM) [23] is an important part of AFF. The local and global features in CNNs are combined by MS-CAM, and the multiscale feature contexts inside the attention

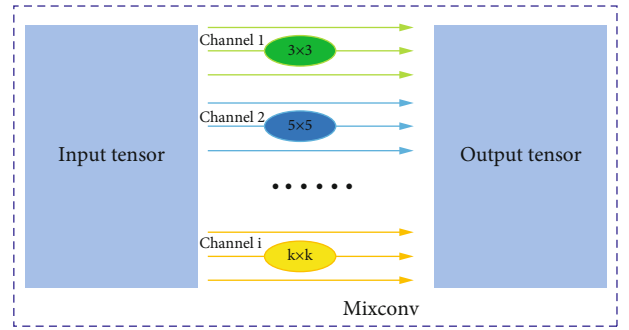


FIGURE 5: MixConv.

module are fused to generate the spatial attention. It poses the scale problem in channel attention and implements it by pointwise convolution instead of different size of kernels. The local and global feature contexts inside the channel attention module are aggregated by MS-CAM. The local channel context can be calculated by equation (4) [23]:

$$L(X) = \mathcal{B}(\text{PWConv}_2(\delta(\mathcal{B}(\text{PWConv}_1(X))))), \quad (4)$$

where PWConv1 has the kernel size of $C/r \times C \times 1 \times 1$, and PWConv2 has the kernel size of $C \times C/r \times 1 \times 1$, \mathcal{B} means Batch Normalization, and δ means activation function.

Global channel context can be calculated as follows [23]:

$$g(X) = \mathcal{B}(\mathbf{W}_2 \delta(\mathcal{B}(\mathbf{W}_1(g(\mathbf{X}))))), \quad (5)$$

where $g(\mathbf{X}) = 1/H \times W \sum_{i=1}^H \sum_{j=1}^W \mathbf{X}_{[:,i,j]}$, which means the global average pooling (GAP).

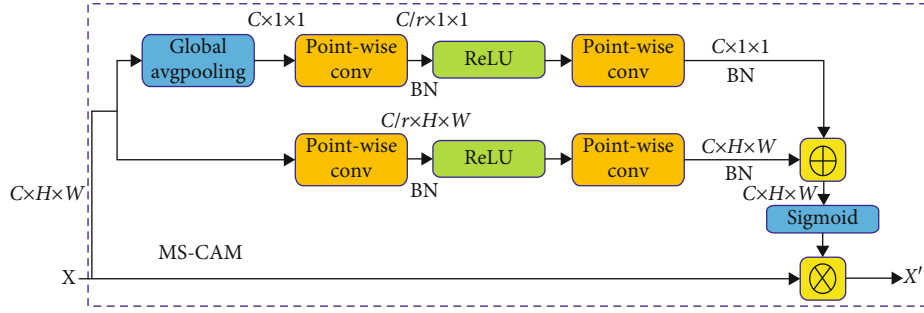


FIGURE 6: MS-CAM.

Assuming global channel context is $g(X)$ and local channel context is $L(X)$, the refined feature $X' \in \mathbb{R}^{C \times H \times W}$ can be obtained by MS-CAM, as shown in the following equation:

$$X' = X \otimes M(X) = X \otimes \sigma(L(X) \oplus g(X)). \quad (6)$$

Here, $M(X) \in \mathbb{R}^{C \times H \times W}$ refers to the attentional weights produced by MS-CAM, \oplus refers to the broadcasting addition, \otimes refers to the element-wise multiplication.

MS-CAM aggregates the multiscale contexts along the channel dimension, thus, simultaneously emphasizing large objects distributed more globally and highlighting small objects distributed more locally. Therefore, MS-CAM can facilitate network identification and detection of objects at extreme scale variations.

Assuming that the two feature maps $X, Y \in \mathbb{R}^{C \times H \times W}$, based on the multiscale channel attention module M , the expression of attentional feature fusion (AFF) is shown in equation (7) [23]:

$$Z = M(X \uplus Y) \otimes X + (1 - M(X \uplus Y)) \otimes Y. \quad (7)$$

The AFF is illustrated in Figure 7, where $Z \in \mathbb{R}^{C \times H \times W}$ is the fused feature, and \uplus is the initial feature integration. The dashed line denotes $1 - M(X \uplus Y)$. It should be noted that the fusion weights $M(X \uplus Y)$ are in the range of 0 to 1, so are the $1 - M(X \uplus Y)$, which enable the network to perform soft selection or weighted averaging between X and Y .

AFF module fuses the features from the same-layer scenario to the cross-layer scenario based on attention, including short and long skip connections, and even performs the initial fusion inside itself. The features of the targets can be obtained with high resolution, thereby improving the recognition accuracy.

4. Dataset and Experiment Setup

4.1. Dataset. Several traffic sign datasets are currently available for public use, such as German Traffic Sign Detection Benchmark (GTSDDB) [47] and Tsinghua-Tencent 100K [9].

In GTSDDB, natural traffic scenes from different types of roads (freeway, highway, rural, and urban) are recorded at daytime and twilight under various weather conditions [2]. The signs are divided in four different categories, i.e., mandatory, prohibitory, danger, and others. However, in GTSDDB,

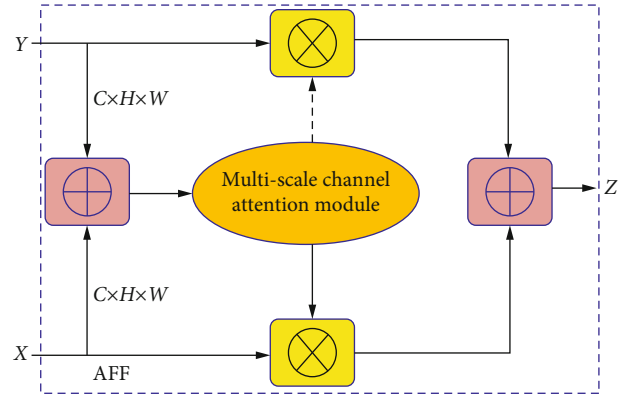


FIGURE 7: AFF.

the traffic sign occupies most of the image, and the algorithms can only determine which subcategory the sign belongs to. Furthermore, no negative samples exist which can disrupt the classification [9]. And most importantly, the four categories are too coarse-grained to determine the exactly accurate traffic sign.

In Tsinghua-Tencent 100K datasets, there are 100,000 images, which contain 30,000 traffic-sign instances. The illuminance and weather conditions in these images are largely variable. The images are collected from Tencent Street Views, covering about 300 Chinese cities and the corresponding road networks [9]. These images are classified into 220 classes (as partly shown in Figure 8), and each class has a unique name. The more fine-grained categorization can help distinguish the traffic sign and give the right instructions to the driver. However, among the 100,000 images, only about 10,000 images contain the traffic signs, and the remaining 90,000 images do not contain any traffic signs. Due to the big dataset, a low percentage (10%) of the images that can contribute to the training, and a large number of negative samples, the training will take a very long time.

To address the above challenges, 1000 additional images were captured by cameras in Shandong, China. In order to focus on analyzing the special weather conditions, these images with traffic signs were taken on rainy, sunny, snowy, and foggy days. In addition, some images contained partly hidden traffic signs. The software named LabelImg was used to annotate the images. For each traffic sign, a bounding rectangle was drawn, and the corresponding category was

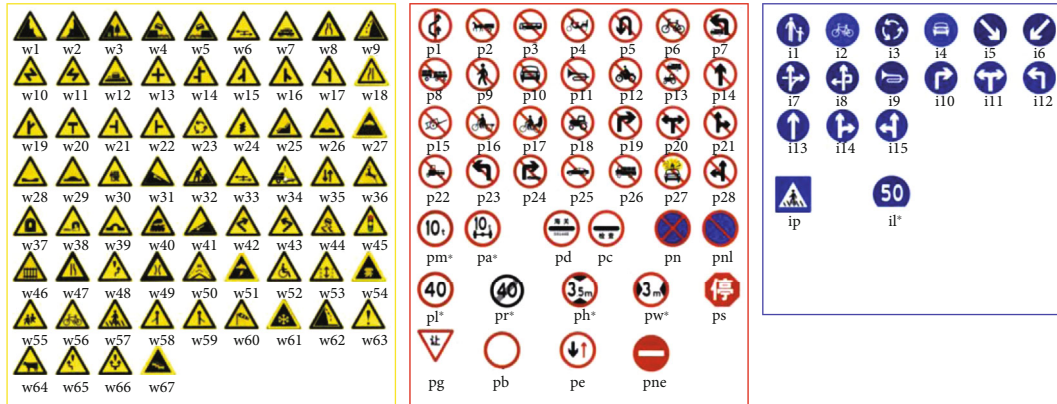


FIGURE 8: Typical traffic sign categories in Tsinghua-Tencent 100K.

manually provided. The annotated image can be automatically saved as an additional JSON file, which contained the top left and bottom right positions of the bounding rectangle in the image and the assigned class. Therefore, the images were annotated and ready for training.

4.2. Data Augmentation. Although there were 10,000 images from Tsinghua-Tencent 100K with traffic signs and 1000 images captured and annotated in Shandong, the training samples were still not sufficient due to the large number (millions) of learnable parameters in the model. To address this issue, we constructed many new samples based on existing images with traffic signs.

In order to create additional synthetic traffic-sign instances, we modified the segmented training samples from the real world [4]. First, the traffic signs were segmented from the annotated images by bounding box, and then the segmented images with only one traffic sign were distorted to simulate more realistic noises. The distortions included Gaussian blurring, resizing, brightness adjustment and contrast adjustment, and adding salt and pepper noises. Finally, the distorted traffic signs were copy-pasted into the Tsinghua-Tencent images without any traffic signs. Using the first two steps, many segmented and distorted traffic signs were ready to use. First, 9 traffic signs were randomly selected for the empty image in Tsinghua-Tencent without any traffic sign. Then, the selected 9 traffic signs were pasted in the empty image one by one. It should be noted that the pasted objects should not overlap with each other, and the distance between the pasted object and the image boundaries should be equal to or larger than five pixels [48]. Figure 9 shows the obtained image with traffic signs by copy-pasting signs in an empty image without any traffic signs.

Additionally, 14,000 augmented images were produced with copy-paste method. Together with the 10,000 images from Tsinghua-Tencent 100K with traffic signs and the 1000 images captured in Shandong, a total of 25,000 annotated images were included in the dataset. In the combined dataset, 5000 samples were randomly selected and used as the test set to assess the performance of the designed network model. The rest 20,000 samples were divided into two groups, i.e., a training group with 15,000 samples and a validation group with 5000 samples.

4.3. Experiment Setup. The experiment was conducted on a CPU Intel Core i7-6700K @4.00GHz, a GPU GTX1080Ti, 12 GB of video memory, the Ubuntu Linux operating system, and a deep learning framework PyTorch 1.6. In addition, Python 3.7 and Open CV 3.41 were used for data augmentation.

The proposed TS-Yolo model was trained using a back-propagation learning algorithm with CIoU (Complete-IoU) as the loss function and the stochastic gradient descent (SGD) as the optimizer. In the training, the learning rate was set to 0.001, the weight decay was 0.0005, and 600 epochs were executed for each training. CIoU Loss [49] contained three parts, i.e., object loss, classification loss, and box loss.

The proposed model was compared with the existing state-of-art one-stage object detection models, such as RetinaNet, SSD, YoloV3, YoloV4, and YoloV5. These models were trained with the above-augmented dataset and validated with the validation dataset.

4.4. Performance Metrics. In this study, the proposed model was evaluated by different metrics. Mean average precision (mAP) is a commonly used metric to evaluate visual object detectors [4]. We used two types of mAPs here, i.e., mAP_{0.5} and mAP_{0.5:0.95}. In both metrics, to be considered as a true positive, the intersection-over-union (IoU) overlap between the detection and the ground truth needs to exceed the defined minimal value. In addition, in order to capture the trade-off between the miss rate and the false-positive rate, the average precision (AP) was computed by calculating the area under the precision-recall curve. AP was calculated independently for each category, and the final metric included the average AP values of all categories. In mAP_{0.5}, the value of IoU overlap was fixed, while in mAP_{0.5:0.95}, the value was the average of the IoU overlap values. The reported values were averaged in 0.05 increment within the IoU overlap range of [0.5:0.95].

Precision and recall values were also used in this study. Precision (P) was used to evaluate how many percentages of the predictions from the results were correct. $P = 100\%$ indicated there were no errors. Recall (R) was used to evaluate how many positive samples were detected correctly. $R = 100\%$ meant there were no missing targets. FPS (frames per second) was also used to evaluate the model inference



FIGURE 9: Empty image and obtained image with signs after the copy-pasting operation.

TABLE 1: The recognition results of different models.

Model	Precision (%)	Recall (%)	mAP_0.5 (%)	mAP_0.5:0.95 (%)	FPS
Faster RCNN	74.12	87.95	85.84	77.95	24
R-FCN	76.69	89.16	87.19	85.28	28
SSD	70.41	76.05	75.11	71.6	80
RetinaNet	69.83	75.71	75.02	71.5	82
YoloV3	70.16	77.4	76.92	72.97	102
YoloV4	69.71	78.5	78.05	73.93	90
YoloV5	71.92	80.31	80.05	75.63	88
TS-YOLO	74.53	84.01	83.73	78.66	83

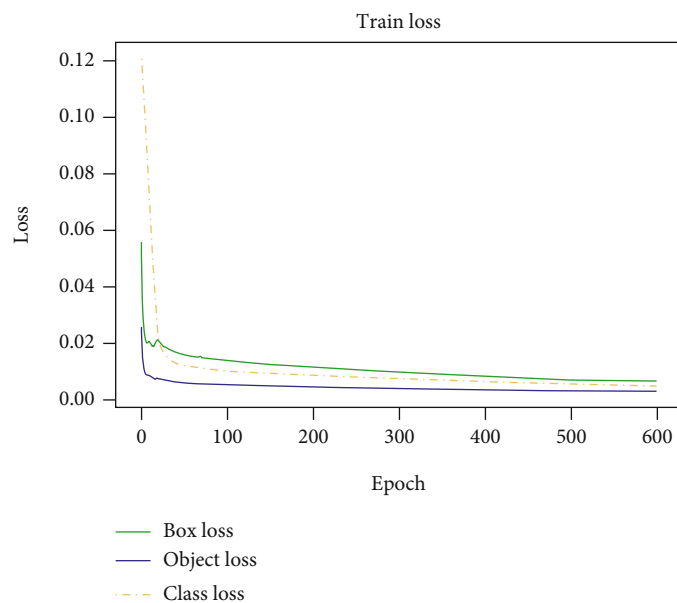


FIGURE 10: Train loss (object loss, class loss, and box loss).

speed, and it indicated that how many frames the trained model could make prediction.

5. Results and Analysis

5.1. *Evaluation on Augmented Tsinghua-Tencent Dataset.* The results of the proposed TS-Yolo and other advanced

methods were obtained on the augmented Tsinghua-Tencent dataset at the above experiment settings. The results are shown in Table 1.

From Table 1, the proposed TS-YOLO model had the best performance in the augmented Tsinghua-Tencent 100K dataset. Using the proposed TS-YOLO model, the precision was 74.53, the recall was 84.01, mAP_0.5 was 83.73,

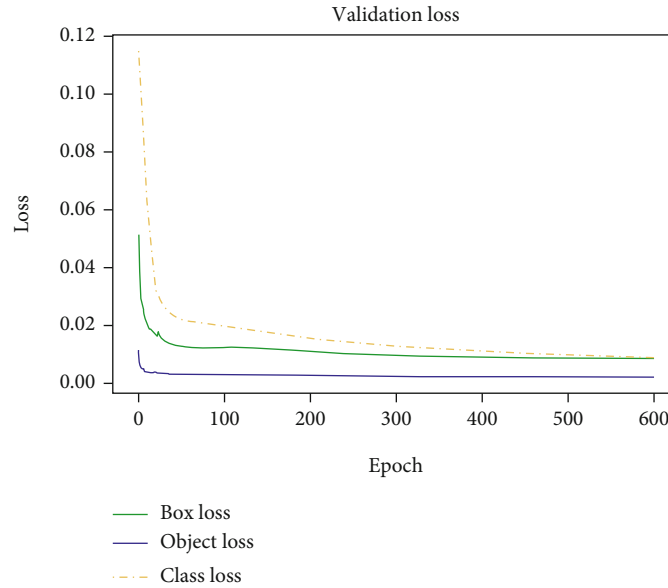


FIGURE 11: Validation loss (object loss, class loss, and box loss).

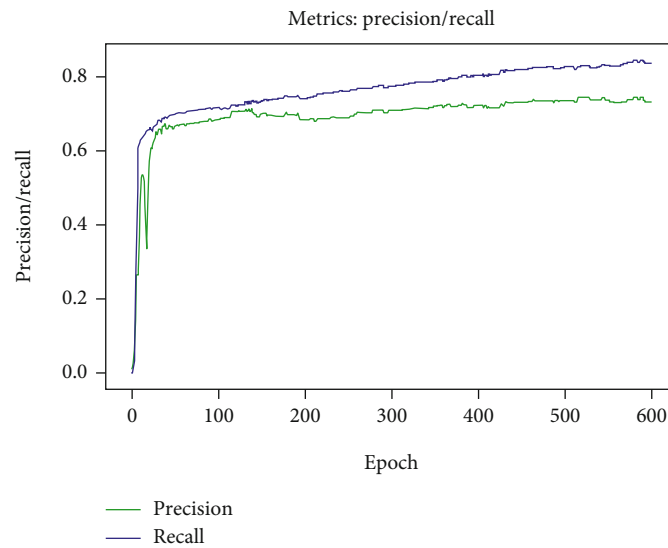


FIGURE 12: Recall and precision during the training.

and $mAP_{0.5:0.95}$ was 78.66. Compared with YoloV5, the precision was increased by 2.61, the recall was increased by 3.7, $mAP_{0.5}$ was increased by 3.68, and $mAP_{0.5:0.95}$ was increased by 3.03, but FPS was decreased by 5% which did not affect too much. Comparing with two-stage detectors, Faster RCNN and R-FCN models, the TS-YOLO model provides only competitive performance in recognition precision; however, in terms of FPS, it is almost three times faster in the inference speed, which is of great significance in real engineering.

More details for TS-YOLO experiments results (training and validation loss) are shown in Figures 10–14 and Table 2.

Figure 10 shows that the loss (object loss, class loss, and box loss) values are large in the initial training stage of the model. However, as the training process progresses, the loss

value shows an overall decreasing trend. The decreasing curve is very smooth, and there are no spikes during all the iterations. The loss stabilized when the training step reached 6000 steps. After finishing the training with 600 epochs, the final training model was obtained.

Figure 11 shows that the validation loss (object loss, class loss, and box loss) is very large in the initial training stage. However, as the training process progresses, the loss value shows an overall decreasing trend. The curves are very smooth, and there are no spikes during all the iterations. When the training was finished at 600 epochs, the loss also reached the minimum, and the trained model had good performance.

The trends of the recall and precision with the training epochs are shown in Figure 12. At the beginning of the

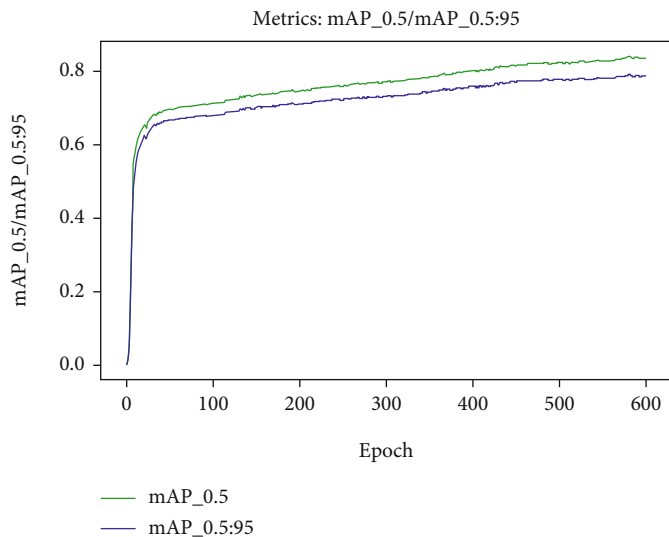


FIGURE 13: mAP_0.5 and mAP_0.5:95 during the training.

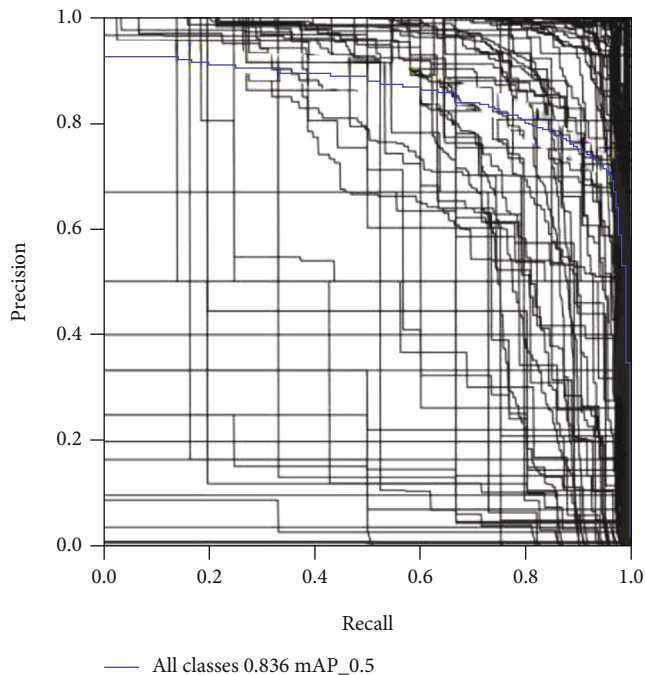


FIGURE 14: PR curve.

training, both recall and precision rapidly increased due to the fast adaption of the model. As the training proceeded, both values started to grow with a stable step, but they still increased gradually. When the training was finished at 600 epochs, the recall and precision also reached the maximum values.

Figure 13 shows the trends of mAP_0.5 and mAp_0.5:0.95 with the training epochs. At the beginning of the training, both mAP_0.5 and mAP_0.5:95 rapidly increased, but mAP_0.5 had a higher value than mAP_0.5:95. As the training proceeded, the training was getting into stable state, in which both mAP_0.5 and mAP_0.5:95 were increased

TABLE 2: Validation results of different traffic sign categories.

Type	Count	<i>P</i>	<i>R</i>	mAP_0.5 (%)	mAP_0.5:0.95 (%)
i1	99	0.994	0.99	98.5	98
i10	130	0.952	0.969	96.5	96.2
i11	100	0.991	1	99.5	99.2
i12	130	0.969	0.992	99.5	99.3
i13	106	0.974	0.991	99.5	99.2
i14	105	0.973	1	99.5	99.5
i15	110	0.993	1	99.5	99.5
i2	188	0.8	0.91	93.9	82.6
i3	114	1	0.99	99.5	99.4
il100	30	0.543	0.9	90.6	71.3
il110	3	0.0941	0.667	68.9	58.2
il50	106	0.978	0.991	99.4	98.8
il60	80	0.711	0.975	95.8	72.5
il80	50	0.588	0.94	88.2	71.8
il90	18	0.403	0.944	73.3	58.2
io	162	0.563	0.883	86	62.1
ip	181	0.797	0.972	97.5	86
p1	124	0.918	0.968	97.4	95.7
p12	143	0.855	0.979	98.2	91.2
p13	126	0.99	0.992	99.5	99.5
p14	121	0.989	0.959	98.2	95.4
p15	124	0.993	0.992	99.5	99.3
p16	98	0.931	0.99	98.5	98.4
p17	140	0.948	1	99.5	98.6

gradually. When the training was finished at 600 epochs, mAP_0.5 and mAP_0.5:95 both reached the maximum values, which indicated the trained model achieved an optimized status.

Table 2 shows some validation results (precision, recall, mAP_0.5, and mAP_0.5:0.95) for different categories of



FIGURE 15: Street view images and detection results from TS-Yolo. ((a) Cloudy; (b) snowy; (c) dimly at night; (d) shadow occlusion).

traffic signs. It can be seen from the table that the model has good performance in precision, recall, mAP_0.5, and mAP_0.5:0.95 for most of the traffic signs. However, some categories such as il100, il110, il60, il80, and il90 had only less than

100 training samples, which were not sufficient for training. Thus, mAP_0.5:0.95 for these categories was less than 0.8. By contrast, p12, p13, p14, and p15 had more training samples, and both mAP_0.5 and mAP_0.5:0.95 had high scores.

TABLE 3: Ablation study results.

Methods	Precision (%)	Recall (%)	mAP_0.5 (%)	mAP_0.5:0.95 (%)
YoloV5 (no data augmentation)	37.36	47.87	46.94	35.09
YoloV5 + data augmentation	71.92	80.31	80.05	75.63
YoloV5 + data augmentation + MixConv	72.27	80.38	80.03	75.7
YoloV5 + data augmentation + AFF	73.88	82.5	81.92	77.4
YoloV5 + data augmentation + MixConv + AFF	74.53	84.01	83.73	78.66

In the PR curve, the recall was plotted on the X axis and precision was plotted on the Y axis. A point in a PR curve represents the corresponding recall and precision values of the classification results at a certain threshold. In detail, at the specific threshold, the model will predict the result greater than the threshold as positive, and the results less than the threshold as negative. The PR curve was generated by moving the threshold from high to low.

Figure 14 shows the PR curve of TS-Yolo with an augmented Tsinghua-Tencent dataset. The blue line shows the precision-recall curve for all the classes when the threshold changes from 0 to 1. The PR curve of all classes is close to the top right corner of the box, which indicates that both precision and recall values are close to 1. Thus, the obtained results of TS-Yolo for classification are relatively good. But for some classes, the PR curve is close to the bottom left corner of the box, which indicates that both precision and recall are close to 0. Thus, the prediction performance for these classes is poor, which is mainly because that these classes don't have enough training samples. For these classes, the learning performance can be improved by adding more training samples.

Figure 15 shows the street views with traffic signs and the detection results from TS-Yolo. These figures show different severe weather conditions. Figure 15(a) shows a road scene in cloudy condition. Figure 15(b) shows a road scene on snowy days, Figure 15(c) shows a dim road scene at night, and Figure 15(d) shows a road scene with shadow occlusion traffic signs. It can be seen from the figures that TS-Yolo performs well and is able to detect all the traffic signs correctly.

5.2. Ablation Study. In order to illustrate the improvements brought by different methods introduced in TS-Yolo, we conducted comparison experiments to assess the effects of data augmentation, MixConv, and AFF.

As shown in Table 3, the results with 10,000 images from the Tsinghua-Tencent traffic sign dataset without any data augmentation were used as the baseline. Using this dataset, YoloV5 had poor performance, the precision was 37.36, and mAP_0.5 was only 46.94. After applying data augmentation, the number of images increased to 25,000, the precision was 71.92, which was increased by 34.56, and mAP_0.5 was 80.05, which was increased by 33.11. When MixConv was applied on YoloV5 with the augmented dataset, the precision was 72.27, which was increased by 0.35, but mAP_0.5 was quite similar. If AFF was applied instead, the precision was 73.88, which was increased by 1.96, and mAP_0.5 was 81.92, which was increased by 1.87. Finally, the TS-Yolo

model was evaluated when all the methods including data augmentation, MixConv, and AFF were applied. In this scenario, the precision was 74.53, which was 2.61 higher than the scenario with data augmentation only, and mAP_0.5 was 83.73, which was 3.68 higher than the scenario with data augmentation only. Therefore, data augmentation contributed most in improving the performance of the model, because high quality and sufficient training data was the most important factor for the model. In addition, the application of MixConv and AFF also improved the accuracy (by 2.61) and mAP (by 3.68), which can benefit the accurate traffic sign detection in severe weather condition.

6. Summary and Conclusions

This paper proposed a CNN-based model, named TS-Yolo, for accurate traffic detection under severe weather conditions. First, the data augmentation was conducted using copy-paste strategy, and a large number of new samples were constructed based on existing traffic-sign instances. Based on YoloV5, MixConv was also used to mix different kernel sizes in a single convolution operation, so that different patterns with various resolutions can be easily captured. AFF module was also used to fuse features from the same-layer scenario to cross-layer scenarios based on attention, including short and long skip connections, and even perform the initial fusion inside itself. The application of the AFF module contributed to capture the features of the targets with high resolutions. The summary and main findings are as follows:

- (i) With data augmentation, the number of images was significantly increased. The copy-pasting operation was used to paste the distorted traffic signs to the street view images without any traffic sign. Thus, a large amount of annotated training samples were generated. After data augmentation, the precision was 71.92, which was increased by 34.56, and mAP_0.5 was 80.05, which was increased by 33.11
- (ii) Based on YoloV5, MixConv was used in backbone to mix different kernel sizes in a single convolution operation, so that the patterns with different resolutions can be captured. AFF was used in neck layer to fuse the features from the same-layer scenario to cross-layer scenarios based on attention, including short and long skip connections, and even perform the initial fusion inside itself. When both MixConv and AFF were in TS-Yolo model, the precision was 74.53, which was 2.61 higher than the test scenario

with only data augmentation, and mAP_{0.5} was 83.73, which was 3.68 higher than the test scenario with only data augmentation

- (iii) With the augmented Tsinghua-Tencent 100K dataset (including images under extreme conditions), the accuracy of traffic signs detection under extreme conditions is improved, which will eventually bring benefits to intelligent driving and transportation systems and help reduce the traffic accident rate and improve transportation safety

Overall, to the best of our knowledge, this research is the first attempt to employ a CNN-based model to be a tentative solution of traffic sign recognition and recognition in low visibility and complex visual environment. The TS-Yolo model integrates into MixConv, and the AFF module can improve the recognition accuracy of traffic signs in extreme adverse environment to a certain extent, which will provide a possible solution for automatic driving and intelligent recognition

7. Suggestions for Future Research

Suggestions for future research on this topic are as follows:

- (i) In the future, more real-world traffic sign images should be captured and annotated to enrich the data set. More abundant data platforms and datasets should be established, and the detection and recognition should be analyzed in various scenarios, such as dust, haze, and tunnel environment. An attempt should be made to use images from video resource so that the images are consecutive frames from it
- (ii) Other CNN object detection models, such as CenterNet and EfficientDet, should be tried to deepen the adaptability of extended model. Furthermore, based on the attention mechanism, a new image recognition architecture should be constructed, so that the implementation of the architecture depends more on the image feature extraction, and then a series of attention guidance modules can be introduced to improve the image recognition accuracy under more extreme conditions
- (iii) The integration of different driving behaviors and traffic sign recognition technology should be studied through experiments and field research, and the basic integration of image acquisition hardware and transmission equipment should be considered to achieve the high integration of automatic driving technology

Data Availability

The (<https://github.com/wanhaifengytu/TSYolo/tree/main/data/trafficsigns>) data used to support the findings of this study have been deposited in the Git Hub repository (<https://github.com/wanhaifengytu/TSYolo>).

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors' Contributions

Haifeng Wan did the conceptualization, methodology, investigation, and writing—original draft. Lei Gao did the conceptualization and methodology. Manman Su did the conceptualization, investigation, and data curation. Qinglong You did the methodology. Hui Qu did the conceptualization and methodology. Qirun Sun did the investigation and writing—review and editing.

Acknowledgments

The authors are grateful for the financial support from the Natural Science Foundation of Shandong Province (Grant No. ZR2020ME238), Shandong Provincial Highway and Traffic Safety Key Laboratory Fund Project (TM18H46), and Yantai Science and Technology Innovation Development Plan Project (2020XDRH104).

References

- [1] M. Gao, C. Chen, J. Shi, C. S. Lai, Y. Yang, and Z. Dong, "A multiscale recognition method for the optimization of traffic signs using GMM and category quality focal loss," *Sensors*, vol. 20, no. 17, article 4850, 2020.
- [2] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, 2018.
- [3] C. Sun, Y. Ai, S. Wang, and W. Zhang, "Dense-RefineDet for traffic sign detection and classification," *Sensors*, vol. 20, no. 22, p. 6570, 2020.
- [4] D. Tabernik and D. Skocaj, "Deep learning for large-scale traffic-sign detection and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1427–1440, 2020.
- [5] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. di Stefano, "Traffic sign detection via interest region extraction," *Pattern Recognition*, vol. 48, no. 4, pp. 1039–1049, 2015.
- [6] W. A. Haque, S. Arefin, A. S. M. Shihavuddin, and M. A. Hasan, "DeepThin: a novel lightweight CNN architecture for traffic sign recognition without GPU requirements," *Expert Systems With Applications*, vol. 168, article 114481, 2021.
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.
- [8] B. Wei, K. R. Hao, L. Gao, and X.-S. Tang, "Detecting textile micro-defects: a novel and efficient method based on visual gain mechanism," *Information Sciences*, vol. 541, pp. 60–74, 2020.
- [9] Z. Zhu, D. Liang, X. Huang, B. Li, and H. Shimin, "Traffic-sign detection and classification in the wild," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2110–2118, 2016.

- [10] R. Girshick, "Fast RCNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.
- [11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [12] R-FCN and J. Dai, "Object detection via region-based fully convolutional networks," 2016, <https://arxiv.org/pdf/1605.06409.pdf>.
- [13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.
- [14] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, Honolulu, HI, USA, 2017.
- [15] P. Wang, L. Zhu, Q. Zhu et al., "An application of back propagation neural network for the steel stress detection based on Barkhausen noise theory," *NDT & E International*, vol. 55, pp. 9–14, 2013.
- [16] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, <https://arxiv.org/abs/1804.02767/>.
- [17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: optimal speed and accuracy of object detection," 2020, <https://arxiv.org/abs/2004.10934/>.
- [18] <https://github.com/ultralytics/yolov5/>.
- [19] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot Multi-Box detector," *Computer Vision—ECCV*, vol. 9905, pp. 21–37, 2016.
- [20] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, Venice, Italy, 2017.
- [21] G. Rescio, A. Leone, and P. Siciliano, "Supervised machine learning scheme for electromyography-based pre-fall detection system," *Expert Systems with Applications. Volume*, vol. 100, no. 15, pp. 95–105, 2018.
- [22] M. Tan and Q. Le, "MixConv: mixed depthwise convolutional kernels," *BMVC*, 2019, <https://arxiv.org/abs/1907.09595?source=techstories.org/>.
- [23] Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional feature fusion," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3560–3569, 2021.
- [24] L. Zhou and Z. Deng, "LIDAR and vision-based real-time traffic sign detection and recognition algorithm for intelligent vehicle," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 578–583, 2014.
- [25] H. Li, F. Sun, L. Liu, and L. Wang, "A novel traffic sign detection method via color segmentation and robust shape matching," *Neurocomputing*, vol. 169, pp. 77–88, 2015.
- [26] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, H. Gómez-Moreno, and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines," *IEEE Transactions on intelligent transportation systems*, vol. 8, no. 2, pp. 264–278, 2007.
- [27] J. M. Lillo-Castellano, I. Mora-Jiménez, C. Figuera-Pozuelo, and J. L. Rojo-Álvarez, "Traffic sign segmentation and classification using statistical learning methods," *Neurocomputing*, vol. 153, pp. 286–299, 2015.
- [28] W. Li, D. Li, and S. Zeng, "Traffic sign recognition with a small convolutional neural network," in *IOP Conference Series: Materials Science and Engineering*, vol. 688, 2019no. 4.
- [29] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: an analysis of spatial transformers and stochastic optimisation methods," *Neural Networks*, vol. 99, pp. 158–165, 2018.
- [30] K. T. Islam and R. G. Raj, "Real-time (vision-based) road sign recognition using an artificial neural network," *Sensors*, vol. 17, no. 4, p. 853, 2017.
- [31] G. Villalonga, J. Van de Weijer, and A. M. López, "Recognizing new classes with synthetic data in the loop: application to traffic sign recognition," *Sensors*, vol. 20, no. 3, p. 583, 2020.
- [32] B. Wei, K. Hao, L. Gao, X. Tang, and Y. Zhao, "A biologically inspired visual integrated model for image classification," *Neurocomputing*, vol. 405, pp. 103–113, 2020.
- [33] Z. Jiang, D. Mallants, L. Peeters, L. Gao, C. Soerensen, and G. Mariethoz, "High-resolution paleovalley classification from airborne electromagnetic imaging and deep neural network training using digital elevation model data," *Hydrology and Earth System Sciences*, vol. 23, no. 6, pp. 2561–2580, 2019.
- [34] B. Wei, H. He, K. R. Hao, L. Gao, and X.-S. Tang, "Visual interaction networks: a novel bio-inspired computational model for image classification," *Neural Networks*, vol. 130, pp. 100–110, 2020.
- [35] H. Nguyen, "Fast traffic sign detection approach based on lightweight network and multilayer proposal network," *Journal of Sensors*, vol. 2020, Article ID 8844348, 13 pages, 2020.
- [36] O. Belghaoui, W. Handouzi, and M. Tabaa, "Improved traffic sign recognition using deep ConvNet architecture," *Procedia Computer Science*, vol. 177, pp. 468–473, 2020.
- [37] T. Yang, X. Long, A. K. Sangaiah, Z. Zheng, and C. Tong, "Deep detection network for real-life traffic sign in vehicular networks," *Computer Networks*, vol. 136, pp. 95–104, 2018.
- [38] S. Song, Z. Que, J. Hou, S. du, and Y. Song, "An efficient convolutional neural network for small traffic sign detection," *Journal of Systems Architecture*, vol. 97, pp. 269–277, 2019.
- [39] D. A. Alghmgham, G. Latif, J. Alghazo, and L. Alzubaidi, "Autonomous traffic sign (ATSR) detection and recognition using deep CNN," *Procedia Computer Science*, vol. 163, pp. 266–274, 2019.
- [40] Á. Arcos-García, M. Soilán, J. A. Álvarez-García, and B. Riveiro, "Exploiting synergies of mobile mapping sensors and deep learning for traffic sign recognition systems," *Expert Systems With Applications*, vol. 89, pp. 286–295, 2017.
- [41] K. Zhou, Y. Zhan, and D. Fu, "Learning region-based attention network for traffic sign recognition," *Sensors*, vol. 21, no. 3, p. 686, 2021.
- [42] S. Pei, F. Tang, Y. Ji, J. Fan, and Z. Ning, "Localized traffic sign detection with multi-scale deconvolution networks," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 355–360, Tokyo, Japan, 2018.
- [43] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.
- [44] T. Chaudhari, A. Wale, A. Joshi, and S. Sawant, "Traffic sign recognition using small-scale convolutional neural network," 2020, ICCIP.

- [45] F. Franzen, C. Yuan, and Z. Li, "Traffic sign recognition with neural networks in the frequency domain," *Journal of Physics: Conference Series*, vol. 1576, article 012015, 2020.
- [46] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.
- [47] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: the German traffic sign detection benchmark," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Dallas, TX, USA, 2013.
- [48] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho, "Augmentation for small object detection," 2019, <https://arxiv.org/abs/1902.07296/>.
- [49] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: faster and better learning for bounding box regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 7, pp. 12993–13000, 2020.