Traffic-Sign Recognition Using Deep Learning

Zhongbing Qin(✉) and Wei Qi Yan(✉)

Auckland University of Technology, Auckland, New Zealand
{xyp1014,wyan}@aut.ac.nz

Abstract. Traffic-sign recognition (TSR) has been an essential part of driver-assistance systems, which is able to assist drivers in avoiding a vast number of potential hazards and improve the experience of driving. However, the TSR is a realistic task that is full of constraints, such as visual environment, physical damages, and partial occasions, etc. In order to deal with the constrains, convolutional neural networks (CNN) are accommodated to extract visual features of traffic signs and classify them into corresponding classes. In this project, we initially created a benchmark (NZ-Traffic-Sign 3K) for the traffic-sign recognition in New Zealand. In order to determine which deep learning models are the most suitable one for the TSR, we choose two kinds of models to conduct deep learning computations: Faster R-CNN and YOLOv5. According to the scores of various metrics, we summarized the pros and cons of the picked models for the TSR task.

Keywords: Traffic signs · Faster R-CNN · YOLOv5 · CNN · NZ-Traffic-Sign 3K

1 Introduction

Traffic scene understanding is an important topic in the field of computer vision and intelligent systems [20, 21]. Traffic signs effectively assist drivers in the process of driving and keep them driving much safely by informing drivers of road status and potential hazards [1]. TSR as one of the important parts of driver-assistance systems has become much valuable and a lot of relevant research work emerged recently.

Generally, there are two steps in a typical TSR. The first one is to locate and get the information of traffic signs in natural scene images, which is known as traffic sign detection. The second step is to categorize detected traffic signs into the corresponding subclasses, which is known as traffic sign classification, the step is generally completed manually. Although TSR has gained a plethora of popularity in driver assistant systems, there are still numerous difficulties for identifying real-world traffic signs by using computer algorithms due to various sizes of visual objects [2, 18, 19], color deteriorations, and partial occlusions [3].

In order to deal with these obstacles, many approaches and algorithms have been proposed. In the past, TSR mainly relies on traditional algorithms for object detection, the pipeline of traffic sign detection normally utilized hand-crafted features to extract region proposals, and combined classifiers to filter out the negatives. Recently, deep learning methods are emerging, various cutting-edge approaches have been widely applied to

M. Nguyen et al. (Eds.): ISGV 2021, CCIS 1386, pp. 13–25, 2021.
https://doi.org/10.1007/978-3-030-72073-5_2

this area, such as deep convolutional networks (CNNs). CNNs have brought possibility of learning features from an amount of data without preprocessing, which avoids the process of designing hand-crafted features and absorbs generalized features [4]. Besides, CNN has been already set forth as an object classifier in machine learning which has been leveraged on traffic sign classification.

In the development of traffic sign recognition, German traffic-sign detection and classification benchmarks brought in a vast majority of benefits for evaluation across various algorithms, which were not comparable until the release of the benchmarks. German Traffic Sign Detection Benchmark (GTSDB) [5] and German Traffic Sign Recognition Benchmark (GTSRB) [6] presented two public extensive and available datasets, there are a few methods that have achieved high accuracy rate based on these datasets. Besides, other datasets are also available in public recent years, such as LISA traffic sign dataset (LISATSD) [1], Swedish Traffic Signs Dataset (STSD) [7], and Chinese Traffic Sign Dataset (CTSD) [3]. The GTSRB and GTSDB datasets are the most popular ones for recognizing traffic signs, a great deal of methods have been successful. The contributions of this paper are shown as follows:

- In order to effectively recognize New Zealand's traffic signs, we have created a new and realistic traffic-sign benchmark, which contains partial traffic sign because of physical and time limitations. The benchmark is composed of seven classes of traffic signs, various sizes of real-world signs were captured. The distinction of this benchmark is that it covers numerous small-size objects, which cannot be identified in off-the-shelf datasets. We call this benchmark as NZ-Traffic-Sign 3K.
- We conducted an experiment for traffic sign recognition based on the latest deep learning model (YOLOV5) and accomplish a comparison across the proposed algorithms. The evaluation results illustrated the robustness.

Overall, the ultimate goal of this paper is to complete the customized traffic-sign recognition and figure out which state-of-art neural networks are better fit into this project.

2 Literature Review

Traffic sign recognition (TSR) has benefited a large number of realistic applications, such as driver assistance system, autonomous vehicles, and intelligent mobile robots since they have delivered the current state of traffic signs into various systems. However, there are a few difficulties for computers to recognize traffic signs on the roads, which are mainly from two aspects: One is related to the complex traffic scene [8], the other is about unbalanced class frequencies in the datasets [5].

As for the difficulty of real-world traffic scenes, traffic signs are always well designed for drivers to easily read and recognize the signs during the driving time, including vivid colors, strong and bolded words, as well as various specific and simplified shapes, it is a tricky task to design the features combined with contaminated conditions [9]. For example, the conditions are with weak illumination, small-size signs in scenes, partial occlusions, rotations and physical damages. All of these factors will have a huge impact on the performance of computer algorithms to recognize traffic signs.

In terms of the characteristics of benchmarks, there is usually an uneven distribution of data categories. As we known, traffic signs have various types. For instance, the GTSRB includes 43 classes with the lowest frequency rate of 0.5% and the highest frequency rate of near 6% across all classes [10].

YOLO is a refreshingly straightforward and effective model for visual object detection. Firstly, YOLO as a simple convolutional neural network simultaneously predicts multiple bounding boxes and class probabilities. It initially is trained based on full images and the performance is optimized [11]. Secondly, YOLO is extremely fast which can achieve more than twice of the mean average precision (mAP) of other real-time systems [12].

However, YOLO still lags behind advanced detection architectures in accuracy. For example, it has a poor performance in accuracy compared to one of the top detection methods, Faster R-CNN [13]. In 2020, three YOLO versions had been released, including YOLOv4, YOLOv5, and PP-YOLO. While YOLOv4 was released, it was considered as the fastest and most accurate real-time detection model. It inherits the DarkNet and has obtained a distinct AP value (43.5%) on COCO dataset while achieved a fast detection speed on Tesla V100. Compared with YOLOv3, the AP and FPS have been effectively improved. After the release of YOLOv4, YOLOv5 emerged with the implementation process, rather than the use of original DarkNet. YOLOv5 has achieved 140 FPS compared with YOLOv4 under the same Ultralytics PyTorch library.

3 Methodology

TSR is considered with both object detection and classification. It is a real-world application that computer vision techniques are aligned to develop driver assistant system. In practice, the implementation of this task usually confronts with uncertain issues, such as color fading, disorientation, and variations in size and shape. Recently, there is a lot of research work which is available to deal with these problems and provide solutions to boost the performance of TSR.

Traffic signs mainly are categorized into three groups: Regulatory (i.e., general, parking and road user restrictions), Warning (i.e., temporary and permanent), Advisory (i.e., guide and route signs, e.g., street name, community facilities, tourist signs, service signs and general information signs). Although the design of traffic signs followed the dominant trends and international standards, the traffic signs have various shapes and functions. Thus, it is necessary to take the customized dataset into consideration for effectively recognizing traffic signs.

3.1 Data Collection

In this project, we used a 12-megapixel wide-angle camera of iPhone 11 to capture the realistic traffic sign images in Auckland. Due to lower frequent appearance of traffic signs to pedestrians and vehicles, we directly took traffic signs using digital cameras instead of recording videos. The pixels of the images are stored in JPEG format with the resolution 1080×1440. Our dataset (NZ-Traffic-Sign 3K) consists of 3,436 images and 3,545 instances in total: Stop (236 samples), Keep Left (536 samples), Road Diverges

(505 samples), Road Bump (619 samples), Crosswalk Ahead (636 samples), Give Way at Roundabout (533 samples), and Roundabout Ahead (480 samples) as shown in Fig. 1.

Fig. 1. The examples of seven classes of traffic signs in our dataset

In order to avoid overfitting during training the chosen models, we utilize data augmentation to expand our dataset. The basic manipulations for data augmentation include flipping, rotating, shearing, and adding noises as well as blurring images. In this case, we merely applied two augmentation operations, including adding noises and blurring images, based on our original dataset because these methods could deal with the distorted objects, which could impact the quality of our dataset and even degrade the accuracy of our training models. The manipulations were implemented by importing a Python library, named Skimage.

3.2 Research Design for Training Faster R-CNN

In this experiment, we chose Faster R-CNN to conduct recognition of traffic signs with on our own dataset. Faster R-CNN needs a traditional CNN as the basic convolutional layers for feature extraction. A pretrained VGG16 model was employed to assist us in exporting the feature map.

In order to successfully implement Faster R-CNN, our dataset follows the structure of PASCAL VOC. The dataset structure is split into five parts, namely, Annotations, ImageSets, JPEGImages, and SegmentationClass as well as SegmentationObject.

Table 1. The parameters for training Faster R-CNN

Parameters	Setting
Momentum	0.9
Learning rate	0.01
Max epochs	200
Batch size	24
Weight decay	0.0005

Due to the implementation based on Python, the dependencies should be preinstalled to setup the experimental environment. *Caffe* must be built with the support of Python layers. The Python packages are needed, including Cython, Python-OpenCV and EasyDict, etc. In order to train Faster R-CNN with VGG16, the CUDA device with Tesla V100-SXM2-16GB are necessitated. Amid training the Faster R-CNN, critical parameters are preliminarily set, and the details are shown in Table 1.

3.3 Research Design for Training Faster YOLOv5

The second model in this project is YOLOv5, which was less than 50 days later than the release of YOLOv4. Although the appearance has gained a lot of attentions and debates in the community, it was indeed published with a number of improvements and distinctions. The improvements are mainly reflected in two aspects: Improved the accessibility for detecting real-time objects and the performance of prediction based on either training speed or accuracy.

In order to train YOLOv5 model, the first step is to label the images in our dataset. A graphical image annotation tool (*LabelImg*) was employed to label the images in our dataset. After generated the label files based on our dataset, the next step is to organize directories which save the training and validation images and labels. The model structure of YOLOv5 is as same as the single-stage object detector. It has three main parts: Model backbone, model neck, and model head.

The choice of activation functions is vital in deep neural network. Recently, there are a lot of activation functions available like Leaky ReLU (LReLU) [14], mish, etc. The chosen activation functions in YOLOv5 are LReLU and sigmoid. Specifically, the LReLU is added into the middle/hidden layers, the sigmoid function is added into the final detection layer. In terms of ReLU, it was proposed to alleviate potential problems caused by zero gradient, which allows a small and non-zero gradient if the unit is not active [14],

$$h^{(i)} = \max\left(w^{(i)T}x, 0\right) = \begin{cases} w^{(i)T}x & w^{(i)T}x > 0 \\ 0.01w^{(i)T}x & else \end{cases} \tag{1}$$

where $w^{(i)}$ represents the weight vector of the i^{th} middle layer and x is the input. For the optimization function in YOLOv5, we have two options, including Stochastic Gradient Descent (SGD) and ADAM. The default optimizer is SGD, which is transferred to ADAM by using the parameter option "-- adam".

In YOLOv5, the loss is computed based on three values: Objectiveness score, class probabilities, and the regression score of bounding box. YOLOv5 imports the Binary Cross-Entropy with Logits Loss (BCELoss) from PyTorch for calculating the compound loss. This method combines a sigmoid layer with the BCELoss in one single class, which is more numerically stable than adding the BCELoss after a sigmoid layer. The unreduced loss is described as:

$$l(x, y) = L = \{l_1, \ldots, l_N\}^T \tag{2}$$

$$l_n = -W_n\left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))\right] \tag{3}$$

where N is the batch size. If the reduction is not zero, the error of a reconstruction is measured by using

$$l(x, y) = \begin{cases} mean(L), & reduction = 'mean' \\ sum(L), & reduction = 'sum' \end{cases} \tag{4}$$

Whilst predicting the multilabel classification, the loss is expressed as follows, which achieves by adding weights into positive instances.

$$l_c(x, y) = L_c = \{l_{1,c}, \ldots, l_{N,c}\}^T \tag{5}$$

$$l_{n,c} = -W_{n,c}\left[p_c y_{n,c} \cdot log\, \sigma\left(x_{n,c}\right) + \left(1 - y_{n,c}\right) \cdot log\left(1 - \sigma\left(x_{n,c}\right)\right)\right] \tag{6}$$

where c is the class number. For example, $c = 1$ refers to the single label classification and n is the number of the instances in the batch as well as p_c is the weight of positive instances for the class c.

Table 2. The installed dependencies for YOLOv5

Package name	Version
Cython	---
matplotlib	≥3.2.2
numpy	≥1.18.5
Opencv-python	≥4.1.2
Pillow	---
PyYAML	≥5.3
Scipy	≥1.4.1
Tensorboard	≥2.2
Torch	≥1.6.0
Torchvision	≥0.7.0
tqdm	≥4.41.0

Table 3. The parameters for training YOLOv5

Parameters	Setting
Momentum	0.95
Learning rate	0.00128
Max epochs	200
Batch size	16
Weight decay	0.000201
giou	1.2
cls	15.7
cls_pw	3.67
obj	20
obj_pw	1.36

In this section, we introduce how we set up the experimental environment and explicit the parameters of training YOLOv5. Firstly, YOLOv5 was developed. The details of requirements for this project are shown in Table 2. Furthermore, this experiment was conducted based on *Colab* using Tesla *V100-SXM2-16GB*.

In order to comprehensively evaluate the performance of YOLOv5 and Faster R-CNN, six metrics were considered in this TSR, namely, Generalized Intersection over Union (GIoU), the predicted probability of Objectness, Classification, Precision and Recall as well as mean Average Precisions with multiple IoU (Table 3).

4 Results

4.1 Experiment Results of Faster R-CNN

In this experiment, we used Faster R-CNN as the detector and VGG16 as the classifier to perform the TSR. The experimental results are provided in Table 4. We evaluate the performance of Faster R-CNN with VGG16 by mainly using three measures, namely, precision, recall, and mean average precision with IoU 0.5. Fortunately, the accuracy of predictions is relatively high across seven classes (Table 5).

In order to evaluate the performance of our proposed model based on smaller traffic signs, we conducted another experiment and justified the results from this perspective. The same measures are applied to estimate the prediction results.

After trained 200 epochs, a trend of convergence is shown in the process of training and validation for the losses of GIoU, Objectness, and classification. In terms of GIoU loss, the final score converges to less than 0.02. Incorporating the GIoU loss improved the model performance based on our datasets [15]. The objectiveness loss is 0.005 during the training and reaches to zero while validating the model. Classification loss almost

Table 4. Experimental results for Faster R-CNN across seven classes

Classes	Precision	Recall	mAP@0.5
Roundabout ahead	0.957	0.952	0.961
Stop	0.970	0.959	0.972
Keep left	0.899	0.903	0.900
Road bump	0.925	0.930	0.933
Crosswalk ahead	0.937	0.939	0.943
Road diverges	0.929	0.930	0.932
Give way at roundabout	0.964	0.958	0.962

Table 5. Prediction results of various sizes of the traffic signs based on Faster R-CNN

Pixel size	Precision	Recall	mAP@0.5
$\leq$200	0.907	0.914	0.915
[200, 400]	0.977	0.973	0.979
$\geq$400	0.945	0.947	0.950

reaches zero both in the processes of training and validation. The results are shown in Fig. 2 and Fig. 3.

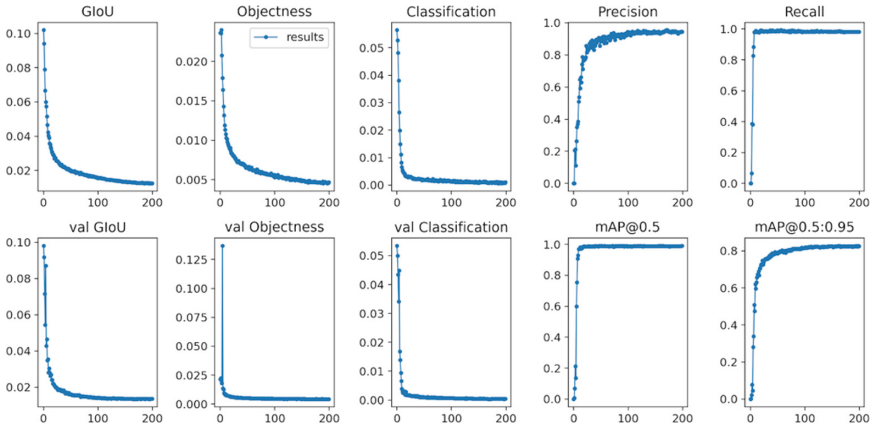

Fig. 2. The loss and precision measures for evaluating the performance of Faster R-CNN

Fig. 3. The test images with class indexes and confidence scores

4.2 Experiment Results of YOLOv5

In this experiment, we chose YOLOv5, a newly released end-to-end network, which is different from the Faster R-CNN. Similarity, we conducted the model training based on the seven classes of our dataset (NZ-traffic-sign 3K). The experimental results are provided in Table 6.

An overall performance of YOLOv5 was justified in this case. The distribution of training and validation sets are invariant, 80% for training and 20% for validation. The evaluation is conducted according to the same measures as the Faster R-CNN, including different losses, precision and recall as well as the mAP with multiple IoU. At the end of the experiment, we test the overall performance of the YOLOv5. The details are shown in Fig. 4 (Table 7).

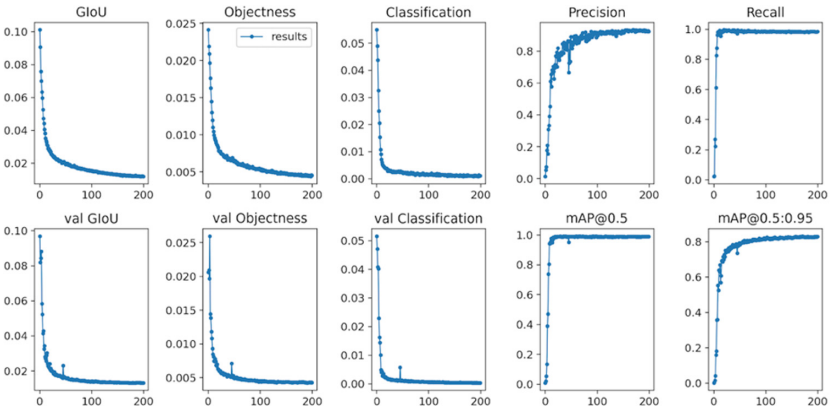

Fig. 4. The losses and precision measures for evaluating the performance of YOLOv5

Table 6. Experimental results for YOLOv5 across seven classes

Classes	Precision	Recall	mAP@0.5
Roundabout ahead	0.949	0.951	0.954
Stop	0.952	0.956	0.959
Keep left	0.901	0.912	0.923
Road bump	0.922	0.927	0.929
Crosswalk ahead	0.933	0.938	0.941
Road diverges	0.934	0.930	0.936
Give way at roundabout	0.955	0.957	0.960

Table 7. Prediction results of various sizes of the traffic signs based on YOLOv5

Pixel size	Precision	Recall	mAP@0.5
$\leq$200	0.883	0.892	0.890
[200, 400]	0.976	0.971	0.974
$\geq$400	0.931	0.939	0.939

5 Analysis

After comparing the results of two models, we concluded that the Faster R-CNN has achieved a higher accuracy rate than YOLOv5 for recognizing the traffic signs in NZ. The Faster R-CNN has achieved lower loss while gaining higher precision, such as the consistent trend of precision and recall as well as mAP.

However, in the testing phase, we noticed that the end-to-end model YOLOv5 is much efficient while it was applied to deal with the data of inference. The test video in the inference is composed of 2,074 frames. The processing time for per frame of the YOLOv5 is only around 0.011 s but the time consumption for Faster R-CNN (37 s) is much longer than YOLOv5. From the perspective of time consumption, YOLOv5 is a much reasonable option for performing the recognition.

In summary, Faster R-CNN is an accurate model for recognizing traffic signs without considering the time consumption. YOLOv5 is a better one if the data processing time is taken into consideration (Fig. 5).

Fig. 5. The test images with class indexes and confidence scores

6 Conclusion and Future Work

One of the objectives of this paper is to propose a customized benchmark for recognizing traffic signs because there is no benchmark that can fit into TSR. Our dataset consists of 3,436 images in total and contains seven classes of traffic signs. The distribution of these classes is more even compared to the most popular benchmark GTSRB, which is an improvement directly contributed to the distinct performance of the two chosen models. Most importantly, we trained CNN models to recognize small traffic signs, thus there are sufficient instances of smaller signs in our dataset. The results of two models based on our dataset are promising and impressive.

Another objective of this paper is to evaluate the neural networks for TSR. We evaluated the performance of a one-stage model (YOLOv5) and a two-stage model (Faster R-CNN with VGG16). According to the comparison between the two models, we see that Faster R-CNN is a better option for TSR without considering the time consumption as the higher-level accuracy. YOLOv5 is much sufficient and important, there is a slightly degrade of accuracy compared to Faster R-CNN.

In future, we will complete our benchmark by covering more types of the traffic signs in NZ so that we can make this project much instructional in this field [16, 17, 27–29]. On the other hand, more object recognition techniques will be employed to TSR [22–26]. For example, recognizing visual objects utilizes heatmaps methods. Finally, more evaluation measures also should be applied to estimate the performance of various models.

References

1. Mogelmose, A., Trivedi, M., Moeslund, T.B.: Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans. Intell. Transp. Syst. **13**(4), 1484–1497 (2012)
2. Zhu, Y., Zhang, C., Zhou, D., Wang, X., Bai, X., Liu, W.: Traffic sign detection and recognition using fully convolutional network guided proposals. Neurocomputing **214**, 758–766 (2016)

3. Yang, Y., Luo, H., Xu, H., Wu, F.: Towards real-time traffic sign detection and classification. IEEE Trans. Intell. Transp. Syst. **17**(7), 2022–2031 (2015)
4. Zhang, J., Huang, M., Jin, X., Li, X.: A real-time Chinese traffic sign detection algorithm based on modified YOLOv2. Algorithms **10**(4), 127 (2017)
5. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. **32**, 323–332 (2012). https://doi.org/10.1016/j.neunet.2012.02.016
6. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German traffic sign recognition benchmark: a multi-class classification competition. In: International Joint Conference on Neural Networks (2011)
7. Larsson, F., Felsberg, M.: Using Fourier descriptors and spatial models for traffic sign recognition. In: Heyden, A., Kahl, F. (eds.) SCIA 2011. LNCS, vol. 6688, pp. 238–249. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21227-7_23
8. Wang, G., Ren, G., Quan, T.: A traffic sign detection method with high accuracy and efficiency. In: International Conference on Computer Science and Electronics Engineering (2013)
9. Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: International Joint Conference on Neural Networks (2011)
10. Mao, X., Hijazi, S., Casas, R., Kaul, P., Kumar, R., Rowen, C.: Hierarchical CNN for traffic sign recognition. In: IEEE Intelligent Vehicles Symposium (IV) (2016)
11. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: IEEE CVPR, pp. 779–788 (2016)
12. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: IEEE CVPR, pp. 7263–7271 (2017)
13. Girshick, R.: Fast R-CNN. In: IEEE ICCV, pp. 1440–1448 (2015)
14. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML (2013)
15. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: a metric and a loss for bounding box regression. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
16. Yan, W.Q.: Computational Methods for Deep Learning - Theoretic. Practice and Applications. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-61081-4
17. Yan, W.Q.: Introduction to Intelligent Surveillance - Surveillance Data Capture, Transmission, and Analytics, 3rd edn. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-319-60228-8
18. Pan, C., Yan, W.Q.: Object detection based on saturation of visual perception. Multimed. Tools Appl. **79**(27–28), 19925–19944 (2020). https://doi.org/10.1007/s11042-020-08866-x
19. Pan, C., Li, X., Yan, W.: A learning-based positive feedback approach in salient object detection. In: IEEE IVCNZ (2018)
20. Liu, X., Yan, W., Kasabov, N.: Vehicle-related scene segmentation using CapsNets. In: IEEE IVCNZ (2020)
21. Liu, X., Neuyen, M., Yan, W.: Vehicle-related scene understanding using deep learning. In: Cree, M., Huang, F., Yuan, J., Yan, W.Q. (eds.) ACPR 2019. CCIS, vol. 1180, pp. 61–73. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-3651-9_7
22. Wang, J., Bacic, B., Yan, W.Q.: An effective method for plate number recognition. Multimed. Tools Appl. **77**(2), 1679–1692 (2017). https://doi.org/10.1007/s11042-017-4356-z
23. Zheng, K., Yan, W., Nand, P.: Video dynamics detection using deep neural networks. IEEE Trans. Emerg. Top. Comput. Intell. **2**(3), 224–234 (2018)
24. Shen, Y., Yan, W.: Blind spot monitoring using deep learning. In: IEEE IVCNZ (2018)
25. Qin, G., Yang, J., Yan, W., Li, Y., Klette, R.: Local fast R-CNN flow for object-centric event recognition in complex traffic scenes. In: Satoh, S. (ed.) PSIVT 2017. LNCS, vol. 10799, pp. 439–452. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92753-4_34

26. Wang, J., Yan, W.: BP-neural network for plate number recognition. Int. J. Digit. Crime Forensics **8**(3), 34–45 (2016)
27. An, N., Yan, W.: Multitarget tracking using Siamese neural networks. ACM TOMM (2021)
28. Liu, X., Yan, W.: Traffic-light sign recognition using Capsule network. MTAP (2021)
29. Xing, J., Yan, W.: Traffic sign recognition using guided image filtering. In: ISGV (2021)