An edge computing-based embedded traffic information processing approach: application of deep learning in existing traffic systems*

PING Haoyu, MA Yongjie, ZHU Guangya, and ZHANG Jiaqi**

School of Physics and Electronic Engineering, Northwest Normal University, Lanzhou 730070, China

(Received 10 November 2023; Revised 3 April 2024)

To address traffic congestion, this study improves MobileNetv2-you only look once version 4 (YOLOv4) target detection algorithm (MobileNetv2-YOLOv4-K++F) and introduces an embedded traffic information processing solution based on edge computing. We transition models initially designed for large-scale graphics processing units (GPUs) to edge computing devices, maximizing the strengths of both deep learning and edge computing technologies. This approach integrates embedded devices with the current traffic system, eliminating the need for extensive equipment updates. The solution enables real-time traffic flow monitoring and license plate recognition at the edge, synchronizing instantaneously with the cloud, allowing for intelligent adjustments of traffic signals and accident forewarnings, enhancing road utilization, and facilitating traffic flow optimization. Through on-site testing using the RK3399PRO development board and the MobileNetv2-YOLOv4-K++F object detection algorithm, the upgrade costs of this approach are less than one-tenth of conventional methods. Under favorable weather conditions, the traffic flow detection accuracy reaches as high as 98%, with license plate recognition exceeding 80%.

Document code: A **Article ID:** 1673-1905(2024)10-0623-6

DOI https://doi.org/10.1007/s11801-024-3247-6

Globally, traffic congestion, accidents, and pollution pose critical challenges[1]. To address traffic safety, many countries have installed high-definition cameras and electronic toll devices on major roads, creating comprehensive traffic management platforms[2,3]. However, the rise of the Internet of Things (IoT), smart transportation, and autonomous driving technologies necessitate deeper analysis of extensive traffic videos and data streams[4,5].

Currently, real-time traffic data is captured using sensors and smart cameras, processed and visualized via deep learning algorithms in the cloud[6-20]. Yet, cloud computing's reliance consumes significant network bandwidth, poses data security risks, and involves high deployment costs. For example, costs at a typical intersection can reach millions of yuan for equipment and network upgrades, impeding the widespread adoption of intelligent transportation systems.

With the rapid development of edge computing and IoT, more cost-effective and innovative solutions for smart city and intelligent transportation are emerging[18]. This study improved MobileNetv2-you only look once version 4 (YOLOv4) target detection algorithm (MobileNetv2-YOLOv4-K++F) and presents an embedded traffic information processing approach based on edge computing. It introduces a novel method for handling traffic data streams without upgrading existing infrastructure. Using an edge computing device, like the Rockchip RK3399Pro (the experimental data for all embedded platform experiments presented in this paper were obtained through testing on the RK3399Pro platform) development board, information from cameras and sensors is processed through pre-deployed deep learning models. This setup enables traffic flow counting, license plate recognition, and vehicle tracking. Processed data is then uploaded to cloud platforms like traffic signal and management systems. The cloud performs functions like intelligent traffic light decision-making and accident alerts based on data from multiple intersections.

In comparison with the conventional approach of extensively replacing expensive intelligent traffic cameras and employing cloud-based computation for uploaded data streams, this approach, integrating edge computing and deep learning, is cost-effective, requiring less than one-tenth of the expenses and minimal network bandwidth. It offers faster processing and enhanced data security, functioning even during network disruptions. Its installation and upgrade costs are low, and it boasts significant scalability. The system can connect to wireless

* This work has been supported by the National Natural Science Foundation of China (No.62066041).

** MA Yongjie is a professor at the School of Physics and Electronic Engineering of Northwest Normal University. He received his Ph.D. degree in 2011 from Lanzhou Jiaotong University. His research interests include evolutionary algorithm, evolutionary neural architecture search, and intelligent transportation image processing. E-mail: myjmyj@nwnu.edu.cn

modules and V2I devices, supporting autonomous driving while ensuring data privacy. This innovative approach facilitates optimized urban traffic flow, assisting in traffic management and enhancing road efficiency.

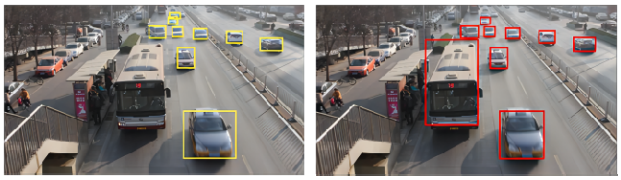
To enhance the performance on embedded devices, we employed MobileNetv2 to replace the backbone and feature extraction network of YOLOv4. The clustering algorithm for anchor box determination was upgraded from the conventional K-means to the K-means++ approach, thereby refining the detection algorithm's accuracy. Furthermore, the incorporation of the focal loss function served to balance the proportion of positive and negative samples, augmenting the precision of the single-stage target detection algorithm. The resultant improved model, referred to as MobileNetv2-YOLOv4-K++F, was trained using consistent parameters and datasets. Performance evaluations were conducted on a personal computer (PC) equipped with a GTX 1650 graphics card, with the detailed outcomes presented in Tab.1, where accurate is obtained at an intersection over union (IoU) threshold of 0.5.

Tab.1 Comparative analysis of algorithm performance with different backbone networks on PC platform

Network model	YOLOv4	Mobile-Netv2-YOLOv4	Mobile-Netv2-YOLOv4-K++F
Accurate (%)	86.40	83.26	87.06
Computational volume (B)	35.48	4.56	4.56
Quantity of participants (M)	63.94	10.38	10.38
Time-consuming reasoning (ms)	194.93	20.72	20.72
Model size (M)	244.00	46.40	46.40

Based on the data presented in Tab.1, it is evident that the enhanced MobileNetv2-YOLOv4-K++F, compared to the original YOLOv4 and MobileNetv2-YOLOv4, achieves the highest detection accuracy under the same model parameters and computational constraints, even surpassing the pre-lightweight YOLOv4 algorithm. The comparative analysis of vehicle detection before and after the algorithmic improvements, as illustrated in Fig.1, clearly demonstrates the improved algorithm's capability to detect a greater number of vehicle targets.

To validate the detection performance of the improved MobileNetv2-YOLOv4 on the RK3399Pro edge computing device, the pre-trained model was first quantized into the reverse K nearest neighbor (RKNN) format using the RKNN Toolkit. Subsequently, it was deployed to the edge device, and the neural processor unit (NPU) was utilized to test the performance of each model. The specific outcomes are presented in Tab.2, where the speed (FPS) represents the results obtained after parallel inference.



(a) Before (b) After

Fig.1 Comparison before and after algorithm improvement

Tab.2 Comparative performance of different backbone networks at the edge

Network model	YOLOv4	Mobile-Netv2-YOLOv4	Mobile-Netv2-YOLOv4-K++F
Time-consuming reasoning (ms)	2 711.02	656.63	73.59
Speed (FPS)	1	5	22
Memory requirements (M)	1 000.00	527.62	274.63
Loading time (s)	93.91	43.37	23.68
Model size (M)	122.16	61.26	10.60

The data from Tab.2 indicate that the enhanced MobileNetv2-YOLOv4-K++F model, in comparison to the quantized YOLOv4, is significantly more compact at only 10.60M in size. It also requires substantially less memory and loading time. The inference speed has increased nearly ninefold, achieving a detection rate of 22 FPS in video processing, which essentially meets the performance requirements for real-time vehicle detection. The experiments confirm that the improved MobileNetv2-YOLOv4-K++F is better suited for deployment on edge computing devices.

The video acquisition module is composed of existing network cameras and toll cameras at the intersection. Through the hardware interface of the data processing module, the data stream is transmitted to the embedded processing platform (Fig.2). The data processing module consists of three components: image preprocessing, vehicle detection and tracking algorithms, and a neural network processing unit. In the image preprocessing stage, video streams are read using OpenCV and Gstreamer to obtain images requiring processing. Subsequent operations include adjusting image dimensions, converting RGB channels to BGR, and grayscale conversion. For object detection, the YOLO detection algorithm is applied to the processed images, identifying the license plate regions. These identified regions are saved and input into the license plate recognition via deep neural networks (LPRNet) character recognition algorithm for precise license plate identification. To ensure continuous and accurate monitoring of each vehicle, the byteTrack tracking algorithm is employed for real-time tracking of every detected license plate. The neural network processing unit can be divided into two phases:

model loading and inferential computing. Initially, the model is deployed on the edge computing device. Subsequently, by invoking the NPU interface, the model is loaded and the runtime environment initialized. This facilitates the execution of deep learning algorithms at the edge, enabling rapid processing of video image data.

Detection and recognition results from the data stream are stored locally in the format license plate number, identification (ID). The saved images of the license plate regions are named in the same manner.

Concurrently, there's real-time communication between local storage and the cloud database. Local data is packaged and uploaded to the cloud in real-time to enrich cloud database resources and facilitate long-term archiving. In contrast, local data undergoes periodic clearance to free up storage space. Once a vehicle is successfully identified, this data is sent to the information integration module, where speed and weight measurements are bound to the vehicle information, recorded in the format license plate number, ID, speed, weight. This integrated information is uploaded in real-time to the traffic information management platform. The platform queries the cloud database for relevant records based on the license plate number to verify the vehicle, simultaneously comparing its speed and weight to determine if there's any speeding or overloading. If any illegal or rule-breaking behavior is detected, the information is uploaded to the violations database, and the responsible party is notified by the database. Moreover, the traffic information management platform aggregates and analyzes data uploaded from the embedded processing platforms at various intersections. It then formulates current traffic light control strategies and releases related warning messages based on this data.

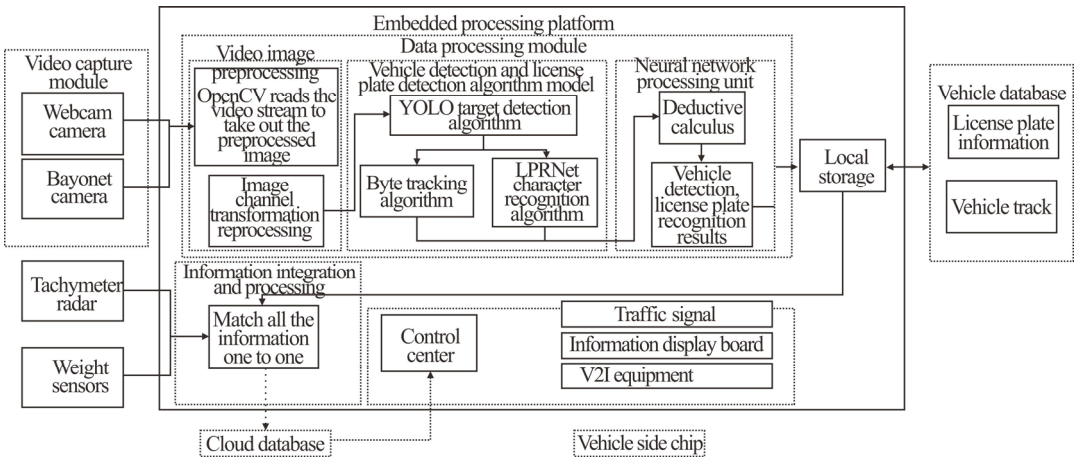


Fig.2 Embedded processing platform

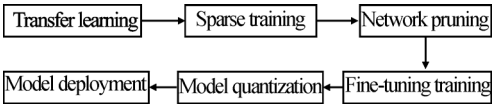


Fig.3 Model compression and deployment

When the embedded processing platform receives commands from the cloud, it controls the traffic lights accordingly and displays relevant notifications and warnings on the information board. Additionally, the embedded processing platform continuously transmits data on traffic flow and vehicle speed to moving vehicles through V2I devices, providing real-time decision-making references for their on-board systems.

To deploy deep learning network models on embedded devices, a series of optimization procedures are currently required. These involve refining models for lightweight efficiency and compressing the network. Only after multiple layers of processing can models be implemented on embedded devices. Model pruning can be classified into structured and unstructured pruning methods. The universality of unstructured pruning is limited, and models produced via this method can only be deployed and applied on specific hardware platforms. To ensure the universality of subsequent experiments, this paper employs structured pruning for model compression. However, if the approach is to be integrated into specific project engineering, a more targeted unstructured pruning should be adopted. The primary steps for model pruning and compression are as follows.

(1) Conduct transfer learning on a specific dataset to obtain the required object detection model.

(2) Sparse the model according to a strategy and post-training, achieving a sparsified network model.

(3) Choose appropriate pruning criteria to prune the sparsified network model.

(4) Fine-tune the pruned model to recuperate any accuracy losses resulting from the pruning process.

(5) Quantify the model to further reduce computational and parameter requirements, enhancing the model's inference performance on edge computing devices.

(6) Deploy the lightweight model to the edge, enabling real-time target detection and various applications.

We utilized the RKNN Toolkit released officially by Rockchip for model deployment. Not only does it support

model conversion, quantization, inference, and performance evaluation, but it also significantly accelerates the deployment and application of deep learning algorithms, simplifying the challenges of deploying deep learning models on edge computing devices. As illustrated in the RKNN development workflow, one first obtains pre-trained models through various deep learning model frameworks. Subsequently, with the assistance of the RKNN Toolkit, we can quantize and convert these pre-trained models, producing RKNN models compatible with the NPU. Ultimately, we can deploy the RKNN models onto edge computing devices and engage in application development by invoking the relevant program interfaces.

We conducted training for the MobileNetv2-YOLOv4-K++F object detection algorithm and the LPRNet character recognition model separately and deployed them on the RK3399PRO development board. The system achieved an average detection speed of over 25 FPS, fulfilling real-time processing requirements. To investigate the detection performance of the system in different environments, we conducted tests under sunny, overcast, and cloudy weather conditions, as shown in Tab.3.

(a) Sunny (b) Cloudy

(c) Overcast

Fig.4 Test site images under different weather conditions

From the presented Tab.3, it can be observed that the accuracy of license plate detection is relatively high under different weather conditions, especially when the weather is favorable and the environmental conditions are suitable, with the license plate detection accuracy approaching 98%. Even in relatively complex conditions, such as cloudy weather, the accuracy of license plate detection can reach 93%. Regarding the accuracy of license plate recognition, overcast weather exhibited the highest recognition rate. This can be attributed to two main reasons. Firstly, in the training of the LPRNet model, the proportion of training data with lighting conditions similar to overcast days is relatively large, resulting in higher recognition accuracy of the model in such environments. Another reason is that in sunny and cloudy weather, the lighting conditions in the license plate area of vehicles are more complex, and the imaging equipment used in the experiment has suboptimal effects on the image processing of local areas, leading to situations such as overexposure or underexposure in the license plate area, thereby affecting the accuracy of license plate recognition.

Tab.3 Test data under different weather conditions

Weather condition	Actual vehicle count	License plate detection count	License plate recognition count	License plate detection accuracy rate	License plate recognition accuracy rate
Sunny	264	264	209	97.67%	79.33%
Overcast	187	182	162	97.33%	86.63%
Cloudy	148	139	103	93.93%	69.63%

To assess the robustness of the system, we repeated tests on a sunny video segment 20 times, and the experimental results are presented in Tab.4.

From the presented Tab.4, it is evident that the license plate detection accuracy is notably high. After 20 consecutive tests, the average accuracy rate reaches an impressive 98% with a variance of only 0.000 27, indicating minimal fluctuations and underscoring the robustness of the model. However, when it comes to license plate recognition, the accuracy rate drops to 79%. Despite this reduction, the model still demonstrates excellent robustness. Two primary reasons account for this decreased recognition accuracy. Firstly, the experiment utilized a standard network camera, which in terms of focal length and clarity significantly differs from cameras specifically designed for license plate capture at checkpoints. Secondly, the LPRNet character recognition model we employed is a public version, which hasn't been deeply trained for specific scenarios, leading to suboptimal recognition performance. Prior to the formal implementation of our solution, we plan to refine the training for both target detection and character recognition models. And by leveraging high-definition camera systems at traffic intersections, we believe there will be a substantial increase in recognition accuracy. However, it's worth noting that under challenging conditions such as rain, snow, or intense/dim lighting, the system's recognition capability maybe compromised, potentially resulting in reduced accuracy. Yet, under stable lighting and environmental conditions, the accuracy of license plate detection and recognition can be significantly improved, achieving or even surpassing 90%.

To tackle imminent traffic challenges, enhance the efficiency and utilization of traffic data streams, break down communication barriers among existing traffic information collection devices, and identify a solution

more cost-effective, efficient, and safe than current strategies, this paper presents an improved MobileNetv2-YOLOv4 target detection algorithm (MobileNetv2-YOLOv4-K++F) and an innovative embedded traffic information processing method based on edge computing. This approach involves pruning and compressing large-scale deep learning models running on GPUs, ultimately deploying them on edge computing devices. This ensures that the detection performance of deep learning models is maximally preserved on compact edge computing devices. By utilizing these edge devices embedded with deep learning models, existing traffic systems can undergo a superior intelligent upgrade at minimal cost. This method seamlessly integrates with the current traffic systems, optimizing the intercommunication and application of traffic data streams. At the edge computing end, it facilitates traffic flow monitoring and license plate recognition, synchronizing with the cloud in real time. This allows for intelligent traffic signal adjustments and accident forewarning, significantly enhancing road efficiency and fluidity. Core advantages of this approach include low costs, high system compatibility, outstanding scalability, and simplified deployment. Importantly, the method substantially conserves network bandwidth and ensures stable operation even during network disruptions. Field validations have shown that by utilizing the RK3399PRO development board combined with the MobileNetv2-YOLOv4-K++F object detection algorithm, the upgrade costs are significantly less than traditional techniques. Under favorable weather conditions, the accuracy rates for traffic flow monitoring and license plate recognition reach 98% and over 80%, respectively.

Although this approach demonstrates substantial potential, it is currently still in the verification phase, necessitating extensive testing to ensure its feasibility in practical applications. Many functionalities within the design of the entire scheme await further validation. Additionally, there is room for refinement and enhancement at various levels of this solution. Particularly concerning small object monitoring and system stability under complex environments, there remains ample scope for improvement.

Tab.4 Results from 20 iterations of license plate detection and recognition on the same test video

Serial number	License plate detection count	License plate recognition count	Actual total license plates	License plate detection accuracy rate	License plate recognition accuracy rate
1	258	204	264	0.977	0.773
2	251	208		0.951	0.788
3	262	210		0.992	0.795
4	258	215		0.977	0.814
5	258	214		0.977	0.811
6	263	213		0.996	0.807
7	260	201		0.985	0.761
8	252	211		0.955	0.799
9	249	212		0.943	0.803
10	262	216		0.992	0.818
11	258	217		0.977	0.822
12	262	212		0.992	0.803
13	260	205		0.985	0.777
14	260	206		0.985	0.780
15	251	205		0.951	0.777
16	255	207		0.966	0.784
17	255	209		0.966	0.792
18	263	210		0.996	0.795
19	259	205		0.981	0.777
20	253	209		0.958	0.792
Variance	-	-	-	0.000 27	0.000 27
Average	257.45	209.45	-	0.975	0.793

Ethics declarations

Conflicts of interest

The authors declare no conflict of interest.

References

[1] YOUNG M S, BIRRELL S A, STANTON N A. Safe driving in a green world: a review of driver performance benchmarks and technologies to support “smart” driving[J]. Applied ergonomics, 2011, 42(4): 533-539.

[2] WEN Y, LU Y, YAN J Q, et al. An algorithm for license plate recognition applied to intelligent transportation system[J]. IEEE transactions on intelligent transportation systems, 2011, 12(3): 830-845.

[3] CHENG Q, MA H T, SUN X. Vehicle LED detection and segmentation recognition based on deep learning for optical camera communication[J]. Optoelectronics letters, 2022, 18(8): 508-512.

[4] GUDIGAR A, CHOKKADI S U R. A review on automatic detection and recognition of traffic sign[J]. Multimedia tools and applications, 2016, 75: 333-364.

[5] ALOMARI A H, ABU LEBDEH E. Smart real-time vehicle detection and tracking system using road surveillance cameras[J]. Journal of transportation engineering, part A: systems, 2022, 148(10): 04022076.

[6] LI J, XU Z J, XU L. Vehicle and pedestrian detection method based on improved YOLOv4-tiny[J]. Optoelectronics letters, 2023, 19(10): 623-628.

[7] LIN H J, YUAN Z L, HE B, et al. A deep learning framework for video-based vehicle counting[J]. Frontiers in physics, 2022, 10: 829734.

[8] DAI Z, SONG H S, WANG X, et al. Video-based vehicle counting framework[J]. IEEE access, 2019, 7: 64460-64470.

[9] UMAIR M, FAROOQ M U, RAZA R H, et al. Efficient video-based vehicle queue length estimation using computer vision and deep learning for an urban traffic scenario[J]. Processes, 2021, 9(10): 1786.

[10] CHEN Y, LU J. A multi-loop vehicle-counting method

under gray mode and RGB mode[J]. Applied sciences, 2021, 11(15): 6831.

[11] BENJDIRA B, KOUBAA A, AZAR A T, et al. TAU: a framework for video-based traffic analytics leveraging artificial intelligence and unmanned aerial systems[J]. Engineering applications of artificial intelligence, 2022, 114: 105095.

[12] AZIMJONOV J, ÖZMEN A, VARAN M. A vision-based real-time traffic flow monitoring system for road intersections[J]. Multimedia tools and applications, 2023: 1-20.

[13] WANG L J, CAO C J, ZOU B H, et al. License plate recognition via attention mechanism[J]. CMC-computers materials & continua, 2023, 75(1): 1801-1814.

[14] ZOU Y J, ZHANG Y J, YAN J, et al. License plate detection and recognition based on YOLOv3 and ILPRNET[J]. Signal, image and video processing, 2022, 16(2): 473-480.

[15] KE X, ZENG G X, GUO W Z. An ultra-fast automatic license plate recognition approach for unconstrained scenarios[J]. IEEE transactions on intelligent transportation systems, 2023.

[16] HE S H, CHEN L, ZHANG S Y, et al. Automatic recognition of traffic signs based on visual inspection[J]. IEEE access, 2021, 9: 43253-43261.

[17] MISHRA J, GOYAL S. An effective automatic traffic sign classification and recognition deep convolutional networks[J]. Multimedia tools and applications, 2022, 81(13): 18915-18934.

[18] DENG Z, LU G M. Intelligent control method of main road traffic flow based on multi-sensor information fusion[J]. Cluster computing, 2023, 26(6): 3577-3586.

[19] AZIMJONOV J, ÖZMEN A. A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways[J]. Advanced engineering informatics, 2021, 50: 101393.

[20] KHAN M M, ILYAS M U, KHAN I R, et al. A review of license plate recognition methods employing neural networks[J]. IEEE access, 2023.