Journal of Smart Science and Technology
2023 September Volume 3 Issue 2
PENERBIT PRESS UNIVERSITI TEKNOLOGI MARA
eISSN: 2785-924X

Design of Automatic Truck Axle Counter Using Deep Learning on NVIDIA Jetson Nano

Rakhmad Gusta Putra[1,*], Wahyu Pribadi[1], Dirvi Eko Juliando Sudirman[1], Muhamad Fajar Subkhan[2]

[1]*Department of Engineering, State Polytechnic of Madiun, Madiun 63133, Indonesia*
[2]*Department of Civil Engineering, State Polytechnic of Malang, Malang 65141, Indonesia*

Received: 28-09-2022
Revised: 11-04-2023
Accepted: 21-06-2023
Published: 30-09-2023

*Correspondence
Email: gusta@pnm.ac.id
(Rakhmad Gusta Putra)

DOI: https://doi.org/10.24191/jsst.v3i2.35

Abstract
To improve the quality of transportation, it is necessary to make right decisions based on accurate data. The flow of vehicles is important in the planning and operation of new roads and the modification of existing roads is needed to meet changes in traffic conditions. To get information about traffic characteristics, various information about traffic infrastructure and driver behavior is needed. The information obtained is analyzed to obtain traffic workload results. If the workload results are below the minimum service standard, a geometric change or arrangement of road space usage is required. The calculation of the number and classification of vehicles has been done manually at certain times. This requires a lot of human resources, so it is less efficient. Vehicle classification is based on the number of axles of the vehicle. This also applies to the payment system on toll roads. Therefore, this research created a real-time truck axle count system using deep learning on the NVIDIA Jetson Nano device. The system must be able to process in real-time with the use of compact hardware and use standard CCTV cameras. Based on the experimental results on 741 test image datasets, the average accuracy is 95.84% with a precision of 91.25%, recall is 79.13% and the processing speed reaches approximately 25 fps on live network camera.

Keywords
Truck; Axle counter; Deep learning; Jetson nano

Citation: Putra, R. J, Pribadi, W, Sudirman, D. E. J, & Subkhan, M. F. (2023). Design of automatic truck axle counter using deep learning on NVIDIA Jetson Nano. *Journal of Smart Science and Technology*, *3*(2), 65-73.

1 Introduction

The transportation sector plays a major role in supporting regional progress. Comparisons of the number of vehicles and the appropriate capacity of the highway is very important. Congestions can arise if the road capacity is not sufficient. These congestions can increase accident rates, harm economic growth, and increase exhaust emissions[1]. The flow of vehicles is an important issue in the planning and operating new roads and modifying roads to adapt to changes in traffic conditions[2]. Various information about traffic characteristics is needed to get various information about traffic infrastructure and driver behavior is needed. The information obtained is analyzed to obtain traffic workload results. If the workload results are below the minimum service standard, a geometric

change or arrangement of road space usage is required.

The calculation of the number and classification of vehicles has been done manually at certain times. This requires a lot of human resources, so it is less efficient. Vehicle classification can be based on the number of axles of the vehicle. This also applies to the payment system on toll roads. Therefore, in this research, a real-time truck axle count system was created using deep learning on the NVIDIA Jetson Nano device. The system must be able to process in real-time with the use of compact hardware, and it can use standard CCTV cameras.

A lot of research has been done to increase the effectiveness of traffic. Among the studies that have been carried out are monitoring and tracking regional density based on the use of WIFI or Bluetooth[3], adaptive traffic lights based on time and day[4], and smart traffic management based on sensor data[5], traffic management based on mechanical and electronic devices[6], using a wireless sensor network[7], traffic density monitoring based on image processing on stationary vehicles[1,8]. Research on comprehensive data retrieval using image processing and vehicle and wheel detection has also been carried out[9] using a novel SSD based[10]. Most studies have not discussed further its implementation and real-time reading capabilities. Image data sets for training and testing use image data collected independently and existing data from previous research[11,12] as presented in Table 2.

This study focuses on the application of the Mobilenet-SSD deep learning model to classify truck types based on the number of axles. The system is applied to the Jetson nano device, a compact mini-PC equipped with an integrated GPU to get the potential for real application in the field.

Many methods are used for the object detection process. The YOLO family, whom has reached version 7, is also very promising. This still needs to be evaluated further to be applied to devices with limited capabilities. The Mobile-Net SSD method is used because it is lightweight for small devices with limited capabilities such as the Jetson Nano. Based on the research benchmark[13] on the Jetson Nano, an Accuracy (mAP) value of 0.26 is obtained when compared to the light YOLO version, YOLOv4-Tiny, which is 0.24. MobileNet SSD latency is 14 milliseconds (ms) while YOLOv4-Tiny is 13 ms. Thus, both the YOLOv4 tiny and the MobileNet SSD are suitable for use in Jetson Nano for real-time applications, with the Mobilenet-SSD having slightly better accuracy with almost the same latency.

2 Methodology

The stages of the research include designing and determining system specifications, experimental design, implementation, testing and analysis. This research was conducted at the Computer Control Engineering Laboratory, Department of Engineering, Madiun State Polytechnic. The test was carried out directly from a network camera installed on the Madiun city ring road, Indonesia. The camera was installed on the side of the road with a height of 180 cm facing the road.

The block diagram of the system and devices used in this study is shown in Figure 1. Image processing device consisting of NVIDIA Jetson Nano, Power Supply, LCD Monitor, network switch, network camera and USB flash drive as the main data storage. The software used includes OpenCV, PyTorch, Tensorflow, TensorRT, and mobileNET SSD as the object detection system. The Jetson Nano is a single board computer (SBC) with Ubuntu OS supported by JetPack 4.2 along with Jetson AGX Xavier and Jetson TX2. their new low-cost AI computer delivers 472 GFLOPS of computing performance to run modern AI workloads and is extremely power efficient, consuming just 10 watts of power. Experiment setup specifications are shown in Table 1. The hardware for the experiment is integrated between the power supply, MCB, main processing unit, display, I/O, and network connection on a 3D printed frame and din rail. The realization of the tool is shown in Figure 2.

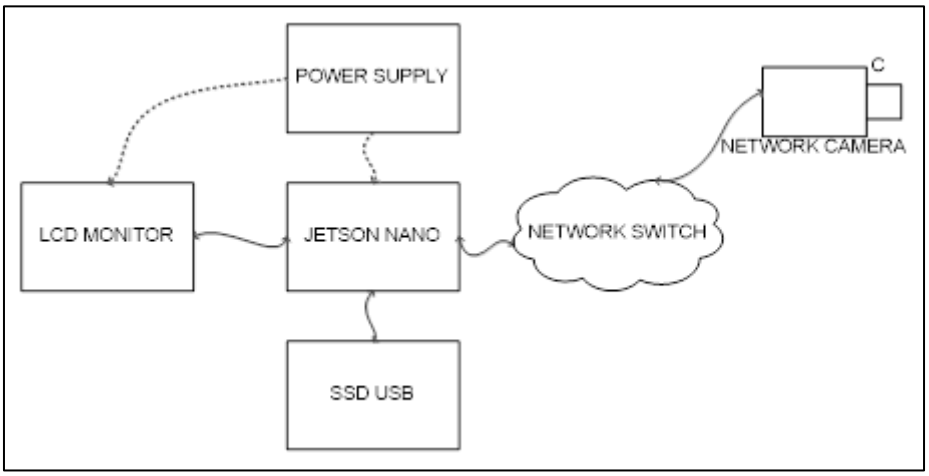

Figure 1. System block diagram.

Table 1. Experiment setup specification.

No	Hardware/ Software	Spesification
1.	Main processing unit	Jetson Nano, 128-core Maxwell GPU, Quad-core Arm A57 processor @ 1.43 GHz, System memory-4GB 64-bit LPDDR4 @ 25.6 GB/s, Storage-microSD card slot
2.	Power supply	5V, 18 A
3.	Display	Portable LCD Display 7”
4.	Input/Output	Wireless mouse and keyboard
5.	Storage	Micro SD card (32 Gb) and USB FDD (32 Gb)
6.	OS	Ubuntu with Jet Pack 4.2

The system algorithm is shown in Figure 2. Data is captured from a camera, static image, or video recording of traffic flow. Because of the main detection is on the axle, the image to be detected must show the axle clearly, as shown in Figure 3. The image criteria used are the vehicle seen from the side, showing the whole vehicle in one frame. If implemented using a CCTV camera, the camera must be able to capture the image of the vehicle from the side.

This followed by pre-processing in the form of taking the Region of Interest (ROI) for the area to be processed. After obtaining the image in the ROI, it is continued with the main processing, object detection using a deep learning model mobileNet SSD algorithm. These algorithms and features allow fast and real-time processing with the features found in Jetson Nano. Detection is focused on four-wheeled trucks or more. Previously, the system had been trained using the existing vehicle dataset. To determine the number of vehicle axles, the vehicle detection results are used as a new ROI (Region of Interest). With a fixed image, it can be assumed that the wheel location is always in the lower area of the detected truck image. the next stage is wheel detection which is carried out in the lower area of the detected truck image. The number of wheels in one vehicle detection is used as a reference for the number of axles of the truck. The calculation results of the truck axle are displayed in the processed image.

MobileNet-SSD consists of the following stages. Pre-processing is carried out using input in the form of images obtained from image capture. This stage aims to adjust the image size to the MobileNet-SSD network architecture. The data type of the image file is ensured in an image format with RGB (Red, Green, Blue) channels. The image is then resized to $300 \times 300 \times 3$. Extraction through the MobileNet Architecture Feature is used as a feature extractor. At this stage the input image will consist of n filter convolution layers. Convolution in the MobileNet

architecture is divided into two, depthwise and pointwise convolution, depthwise convolution is a convolution layer with a kernel size of 3 × 3 with the number of withdrawals or steps for each convolution calculation of 2 pixels and 1 pixel, whereas pointwise convolution has a kernel size of 1 × 1 with a number of locks or strides of 1 pixel[14]. The MobileNet architecture utilizes Batch Normalization (BN) and Rectified Linear Unit (ReLU), BN is used to normalize contributions to layers in each mini group, ReLU is a CNN activation layer by applying the function $f(x) = \max(0,x)$, if $x \leq 0$ then $x = 0$ and if $x > 0$ then $x = x$. The end result of this process is a 14 × 14 image feature with 512 filters. In object classification, classification and localization are carried out simultaneously. The image feature resulting from feature extraction is predicted by object prediction on the image, object detection is carried out using the pre-trained MobileNet model, and the pre-trained will be able to recognize the types of objects present in the image. Object detection is performed on each input image measuring 300 × 300, and feature extraction is carried out by passing through 13 convolution layers so that the final result is 14 × 14. The bounding box prediction is carried out by the SSD architecture where predictions can reach 8732 predictions. Of the many predictions, the non-maximum suppression method eliminates prediction results with a low truth value according to the applied threshold value of 0.45. Then the values obtained from the predictions are the coordinate bounding box, the confidence value bounding box, the class object, and the trust value class object. The end result of object detection is an input image that has detection results with bounding boxes, class beliefs, and class objects.

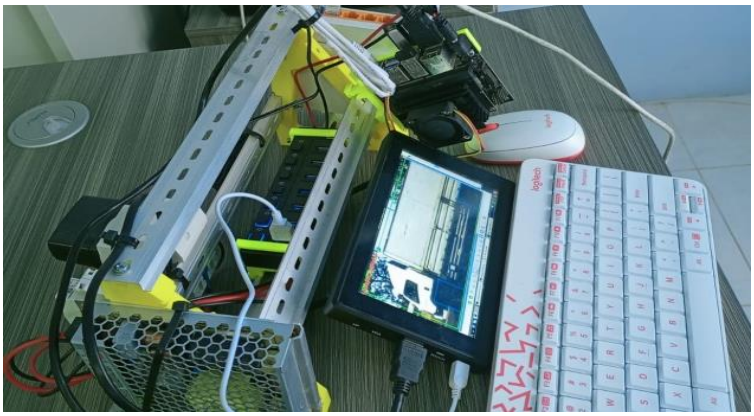

Figure 2. Experiment setup.

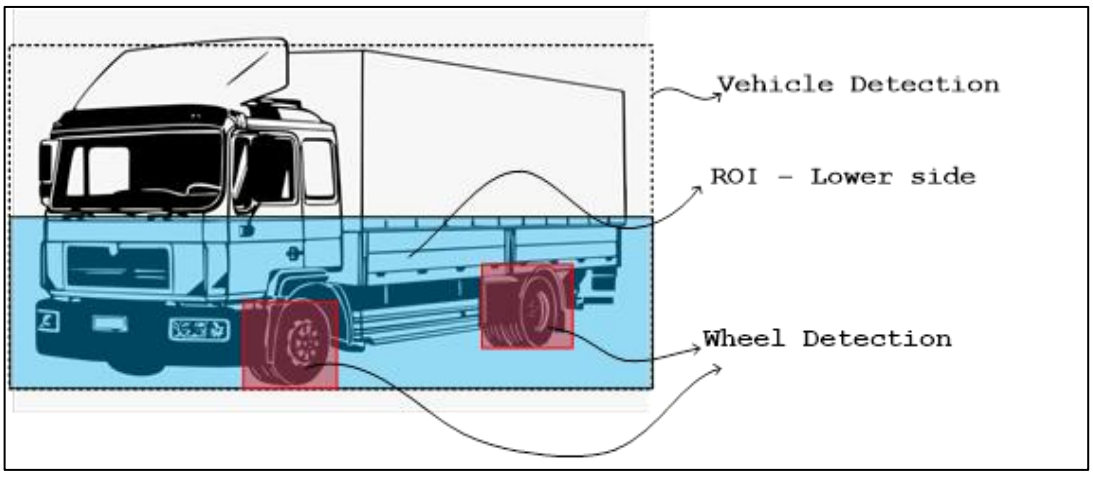

Figure 3. Illustration of wheel axle counting system.

3 Result and Discussion

The experiment was carried out in two stages, the training and testing stages. The training stage included the collection of image datasets obtained from video captures and image datasets available online. Dataset image retrieval was mostly done on the streets of Indonesia with a little additional image datasets from Brazil. References in dataset retrieval have been included in the bibliography[11,12]. In addition to the dataset from existing references, it was also obtained by recording a video using a camera on the highway for several hours. The location of data collection was carried out on the ring road of the Madiun City, Indonesia. The video was then processed manually by taking screenshots for relevant images. An example of an image dataset is shown in Figure 4. Image annotation is divided into two categories, namely trucks and wheel axles. The training image consists of 1481 wheel images and 440 truck images. On 489 images as shown in the Table 3.

The training was carried out on the 4 Gb variant of the Jetson Nano hardware via pyTorch and Jetson Inference with a total of 350 epochs.

The testing stage was carried out by testing the results of the training that has been carried out using a testing dataset of 741 images. Each image detected objects according to the proposed algorithm and calculates the number of wheel axles in the truck object detection area. The number of axles were then calculated to classify the detected trucks as a reference for the number of axles. In addition to the classification of the type of truck, the processing time for one frame was also calculated to obtain the processing speed in frames per second (FPS).

The experimental results were analysed using a confusion matrix. The confusion matrix is a method commonly used to help calculate the accuracy of classification results in object detection. In the confusion matrix, there are 4 (four) terms representing the results of the classification process. The four terms are True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). True Positive (TP) is a condition where the target object is successfully detected by the system; False Positive (FP) is a condition where the system considers another object or condition as the target object; False Negative (FN) is a condition where there is a target object that is not detected by the system; True Negative (TN) is a condition where the system can ignore objects other than the target. To get the values of accuracy, precision, and recall, the following equation was used.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \quad (1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% \quad (3)$$

Table 2. Training and testing dataset.

	Training set	Testing set
Images	489	741
- Axle	1481	
- Truck	440	

Testing was carried out using the NVIDIA Jetson Nano hardware variant 4 Gb with a sample dataset of 741 images. Based on the experiment, the results were shown in Table 4 in percent units. Analysis using a confusion matrix was carried out for each category, namely non-truck, a truck with 2 axles, a truck with 3 axles, a truck with 4 axles, and trucks with 5 axles.

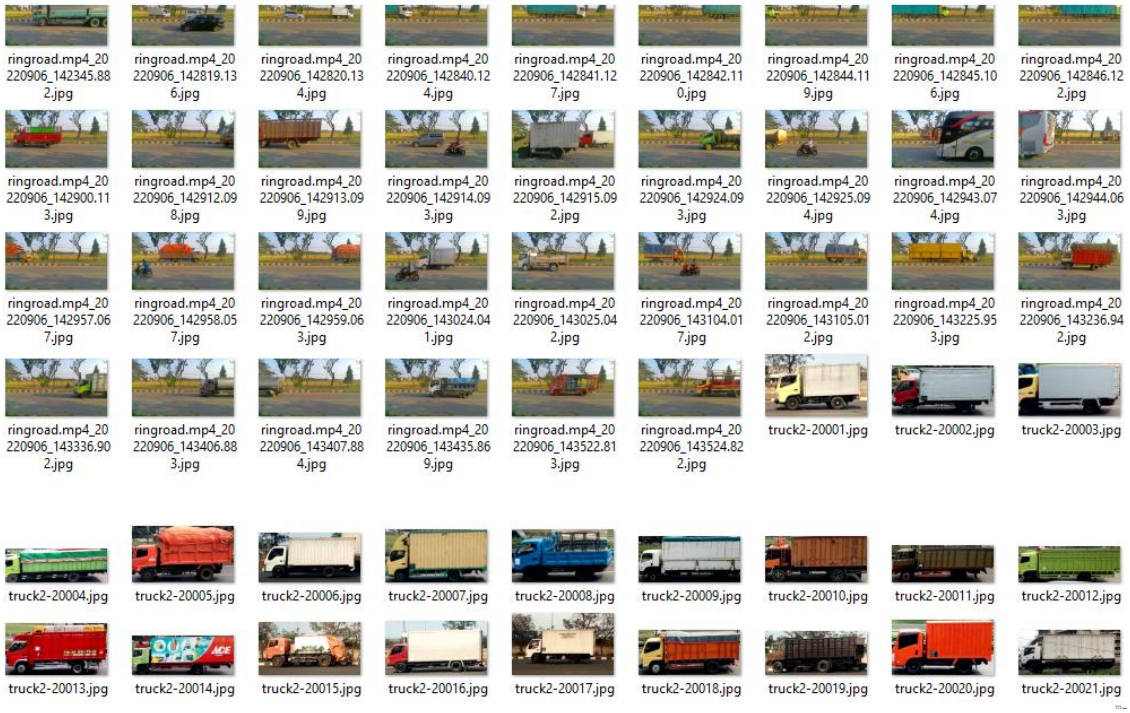

Figure 4. Dataset image samples.

Table 3. Detection and classification results.

SSD MobileNet	True positive	True negative	False positive	False negative
Non-Truck	174	559	0	4
Truck 2-Axle	171	539	24	8
Truck 3-Axle	172	526	37	7
Truck 4-Axle	129	557	5	51
Truck 5-Axle	9	715	1	17

Based on that, these experiments obtained good accuracy and precision with a percentage between 82% to 99% (Figure 5). For the recall value obtained, the lowest value is 34% and the highest is 97%. It can be seen in the results that the more the number of wheel axles of the truck that must be detected, the detection value will decrease because several wheel axles fail to detect. The processing speed obtained is approximately 25 fps which is a speed that can be applied in real-time.

The results of the experiments carried out on the sample images are shown in Figure 6 to Figure 10 below. Non-truck vehicles will not be detected by the system as non-truck vehicles. If the Truck is detected, then the axle in the Region of Interest (ROI) will be detected and calculated. Most of the detections were successful. The failure to detect may be caused by the number of training datasets and testing datasets that are less diverse and numerous, especially for image samples of 5-wheeled trucks or more. Testing the system application and the processing speed is carried out by taking images from a network camera installed on the ring road of the city of Madiun, Indonesia. Compared to vehicle counters with physical sensors, the use of camera sensors and deep learning made installation in the field simpler.

Table 4. Performance of experiment results and processing speed.

SSD MobileNet	Accuracy (%)	Precision (%)	Recall (%)	Processing speed (fps)
Non-Truck	99.45	100.00	97.75	± 25
Truck 2-Axle	95.68	87.69	95.53	
Truck 3-Axle	94.07	82.29	96.08	
Truck 4-Axle	92.45	96.26	71.66	
Truck 5-Axle	97.57	90.00	34.61	
Average	95.84	91.25	79.13	

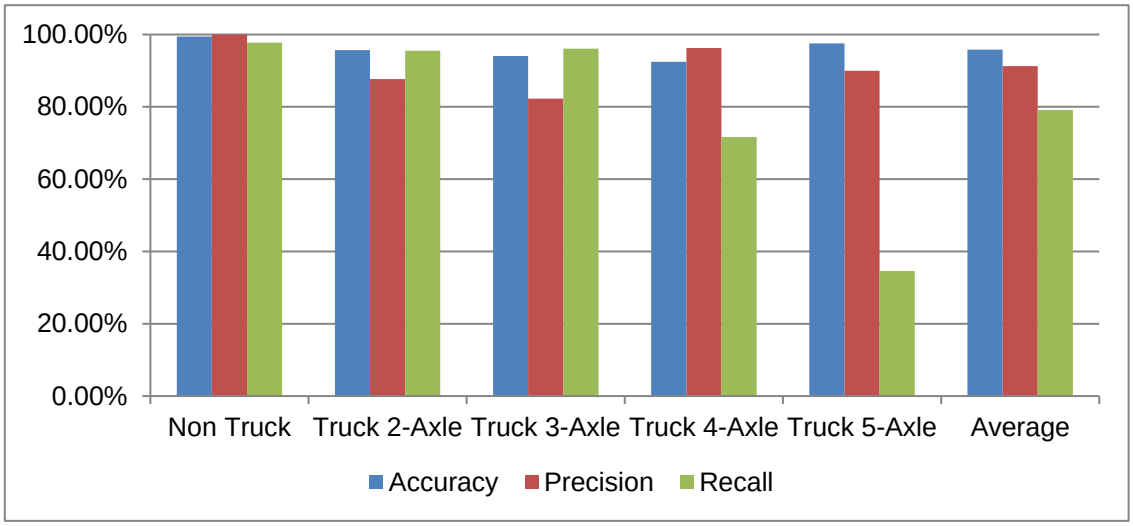

Figure 5. Detection performance chart.

Figure 6. Non-truck detection results.

Figure 7. 2 Axle-truck detection results.

Figure 8. 3 Axle-truck detection results.

Figure 9. 4 Axle-truck detection results.

Figure 10. 5 Axle-truck detection results.

4 Conclusion

Based on the experimental results, it was found that the proposed system has been realized with an average accuracy value of 95.84%, a precision value of 91.25%, and a recall value of 79.13%. The recall value decreases for the larger number of axles. Processing speed reaches approximately 25 fps, which indicates a good processing speed for applications in the field. The performance of the NVIDIA Jetson nano is quite suitable to be used for real-time applications in the field at a relatively affordable price. Future work can be done by increasing the number of datasets for training with various image datasets, using other hardware to compare performance, and using hybrid detection and image classification methods to obtain comparative results. A real-time camera application in the field is needed.

Conflict of Interest

The authors declare that there is no conflict of interest.

Acknowledgment

The authors are grateful to Department of Engineering, State Polytechnic of Madiun, Madiun for the support and assistance.

Funding

This research was funded by State polytechnic of Madiun competitive research program.

Author Contribution

Conceptualization: Putra, R.G., & Subkhan, M.F.
Data curation: Pribadi, W.
Methodology: Putra, R.G., & Sudirman, D.E.J.
Formal analysis: Putra, R.G.
Visualisation: Putra, R.G.
Software: Putra, R.G.
Writing (original draft): Putra, R.G.
Writing (review and editing): Putra, R.G.
Validation: Sudirman, D.E.J., & Pribadi, W.
Supervision: Subkhan, M.F., & Pribadi, W.
Funding acquisition: Putra, R.G.
Project administration: Pribadi, W.

References

1. De Souza, A. M., Brennand, C. A., Yokoyama, R. S., Donato, E. A., Madeira, E. R., & Villas, L. A. (2017). Traffic management systems: A classification, review, challenges, and future perspectives. *International Journal of Distributed Sensor Networks*, *13*(4). https://doi.org/10.1177/1550147716683612
2. Morlok, E. K. (1995). *Pengantar teknik dan perencanaan transportasi* [Introduction to transportation technique and planning]. Erlangga.
3. Fernández-Ares, A., Mora, A. M., Arenas, M. G., García-Sánchez, P., Romero, G., Rivas, V., Castillo, P.A., & Merelo, J. J. (2017). Studying real traffic and mobility scenarios for a Smart City using a new monitoring and tracking system. *Future Generation Computer Systems*, *76*, 163-179. https://doi.org/10.1016/j.future.2016.11.021
4. Yousef, K. M. A., Shatnawi, A., & Latayfeh, M. (2019). Intelligent traffic light scheduling technique using calendar-based history information. *Future Generation Computer Systems, 91*, 124-135. https://doi.org/10.1016/j.future.2018.08.037
5. Finogeev, A., Finogeev, A., Fionova, L., Lyapin, A., & Lychagin, K. A. (2019). Intelligent monitoring system for smart road environment. *Journal of Industrial Information Integration, 15*, 15-20. https://doi.org/10.1016/j.jii.2019.05.003
6. Rath, M. (2018, June). Smart traffic management system for traffic control using automated mechanical and electronic devices. *In IOP Conference Series: Materials Science and Engineering*, *377*, 012201. IOP Publishing. https://doi.org/10.1088/1757-899X/377/1/012201
7. Tubaishat, M., Shang, Y., & Shi, H. (2007, January). Adaptive traffic light control with wireless sensor networks. In *2007 4th IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA (pp. 187-191). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/CCNC.2007.44
8. Putra, R. G., Pribadi, W., Yuwono, I., Sudirman, D. E. J., & Winarno, B. (2021). Adaptive traffic light controller based on congestion detection using computer vision. *Journal of Physics: Conference Series, 1845*(1), 012047. https://doi.org/10.1088/1742-6596/1845/1/012047
9. Zhang, B., & Zhang, J. (2020). A traffic surveillance system for obtaining comprehensive information of the passing vehicles based on instance segmentation. *IEEE Transactions on Intelligent Transportation Systems, 22*(11), 7040-7055. https://doi.org/10.1109/TITS.2020.3001154
10. Fu, J., Zhao, C., Xia, Y., & Liu, W. (2020). Vehicle and wheel detection: A novel SSD-based approach and associated large-scale benchmark dataset. *Multimedia Tools and Applications, 79*, 12615-12634. https://doi.org/10.1007/s11042-019-08523-y
11. Sasongko, A. T., Jati, G., Fanany, M. I., & Jatmiko, W. (2020). Dataset of vehicle images for Indonesia toll road tariff classification. *Data in Brief, 32*, 106061. https://doi.org/10.1016/j.dib.2020.106061
12. Marcomini, L. A., & Cunha, A. L. (2022). Truck axle detection with convolutional neural networks. *arXiv*. https://doi.org/10.48550/arXiv.2204.01868
13. Kang, P., & Somtham, A. (2022). An evaluation of modern accelerator-based edge devices for object detection applications. *Mathematics, 10*(22), 4299. https://doi.org/10.3390/math10224299
14. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv.* https://doi.org/10.48550/arXiv.1704.04861