# Analysis of the vehicle's flow based on the neural network and the SIFT method

*Victor* Ivliev[1], *Evgeniy* Ivliev[1*], *Pavel* Obukhov[1], and *Alexander* Obukhov[1]

[1]Don State Technical University, Gagarin square, 1, Rostov-on-Don, 344000, Russia

**Abstract.** The article presents a vehicle counting system based on TensorFlow neural network models and the SIFT machine vision method. An experimental comparison was made of five detectors consisting of meta-architecture (Faster R-CNN, SSD) and neural networks extracting features (Resnet V1 100, Inception V2, Inception Resnet V2 and Mobilenet V1). The main aspects of these detectors are analyzed, such as accuracy, speed, memory consumption, the number of floating point operations per second and the number of trainable parameters of convolutional neural networks. The calculation of vehicles is carried out by an algorithm based on the SIFT method. This algorithm compares the descriptors of all vehicles in the frame at the current time with the descriptors at the previous time. Based on the maximum match of the descriptors, the algorithm assigns the vehicle identification number from the previous frame, and in the absence of matches creates a new identification number. This approach will make it possible to calculate vehicles more accurately and assess their trajectory and speed.

## 1 Introduction

Traffic jams on motorways are a serious problem all over the world. Motorways are the main roads connecting major cities around the world and designed to accelerate traffic flow. Unfortunately, as congestion increases, travel time increases significantly, and motorways cease to perform their function. Such congestion causes a number of serious problems, including environmental pollution and high accident rates. As a result, time, fuel and money are wasted every year. Building new highways is not always a practical solution to this problem. However, traffic jams can be reduced if alternatives are found that will make existing motorways more efficient. One such alternative is to use the rapid development in computer vision and artificial intelligence to create more sophisticated traffic monitoring and control schemes [1, 2, 3]. Vehicle detection algorithms are an important part of intelligent traffic management systems and operational systems. Therefore, vehicle detection algorithms have attracted a lot of attention and have become a popular area of research in the field of computer vision.

* Corresponding author: 123ivliev123@mail.ru

## 2 Review literature and associated work

Analyzing the research on the detection and assessment of vehicle flows, it is safe to say that the world has stepped far ahead. In the early stages of vehicle detection, the following approaches were used: background subtraction, optical flow methods, probabilistic methods, machine learning, image processing, feature extraction and data aggregation.

To detect vehicles in the research[4], the authors used an algorithm for background subtraction, which showed its efficiency during dynamic camera movement in conditions of intense traffic flow. However, this approach detects well only those vehicles that are in close proximity to the camera, perceives the rest of the vehicles as noise and removes them along with the background.

Some researchers decided to use the original YOLO algorithm [5, 6] for vehicle detection tasks, which provides higher detection accuracy, but also requires high computational costs. Although the YOLOtiny model has relatively small parameters and computational complexity, it has low detection accuracy and cannot effectively detect vehicles.

Vehicle tracking is a key factor for assessing traffic flow. It is thanks to him that it is possible to accurately calculate vehicles, assess their trajectory, as well as their speed.

The study [7] presents an algorithm that tracks an object based on the bounding box of the object. Despite the speed of this tracker, it has disadvantages, since it receives an array of bounding boxes from the detector only with the location of the object. That is, it processes only the coordinates of the object and if the detector fails, the object will be skipped and the tracker will not work reliably. In [8], the authors propose to track the object not only by the coordinate of the bounding box, but also by its shape. However, the same problem remains as in the study [7].

## 3 Materials and Methods

### 3.1 Vehicle detection using neural networks

Vehicle detection is a primary task in the issue of further control of traffic flows. Only after the object is detected, the systems of high-quality and complete data collection in real time, such as traffic flow intensity, traffic directions and average speed, are built-in.

TensorFlow Object Detection API was used to develop neural network models for vehicle detection, which is an open source platform based on Google TensorFlow [9]. Each neural network model consists of a neural network for detecting objects and a neural network extracting features.

Faster R-CNN [10] and SSD [11], which are based on convolutional neural networks (CNN), were considered as a neural network for detecting objects.

The following architectures were selected for feature extraction: Resnet V1 101[12], Inception V2[13], Inception Resnet V2[14] and Mobilenet V2[15].

The experiment consists in training and comparing the results of the effectiveness of various combinations presented in table 1. The experiment was conducted by analogy with the studies [16, 17].

When evaluating efficiency, not only standard accuracy metrics are compared, but also other factors such as memory consumption, the number of floating point operations per second(FLOPS) and operating time, which also play a critical role.

To simplify the experiment, models from the Microsoft COCO dataset were used [18].

**Table 1.** Combinations of meta-architectures for object detection and architectures for feature extraction.

| | Faster R-CNN | SSD |
|---|---|---|
| Resnet V1 101 | ✓ | |
| Inception V2 | ✓ | ✓ |
| Inception Resnet V2 | ✓ | |
| Mobilenet V2 | | ✓ |

To train the neural network, a dataset consisting of 1016 images of vehicles was created. To check the recognition quality, a training sample of 100 images was created. The data annotation was performed by the Labeling program, which helps to highlight the boundaries of the object of interest and specify the class to which this object belongs.

### 3.2 An algorithm for estimating the flow of vehicles

The algorithm for detecting and analyzing information about the flow of transport is shown in Figure 1.

At the **first** stage, a video stream is captured, from which a sequence of two frames is taken: the first frame at time *t* (*frame(t)*), and the second frame at time *t-1*(*frame(t-1)*).

At the **second** stage, a neural network model for detecting vehicles processes the received frames. At the detection stage, the detector also classifies the vehicle.

In the **third** stage, the extraction of areas of interest is performed, i.e. all vehicles detected on frames *frame(t)* and *frame(t-1)*.

At the **fourth** stage, it is necessary to identify key points in the segmented area. It was decided to use the SIFT method for this task. Lowe's SIFT solves the problems of image rotation, affine transformations, intensity, and changing the viewpoint in the corresponding objects. The SIFT algorithm consists of 4 main steps. The first is to estimate the extremes of the scale space using the Gaussian difference (DoG). Secondly, localization of key points, in which candidate key points are localized and refined by eliminating points with low contrast. Thirdly, assigning the orientation of the key point based on the local image gradient and, finally, a descriptor generator to calculate the local image descriptor for each key point based on the magnitude and orientation of the image gradient [19].

At the **fifth** stage, it is necessary to compare an array of descriptors from one frame with an array of descriptors from another frame to obtain information about the direction of movement and the conditional speed of the vehicle.

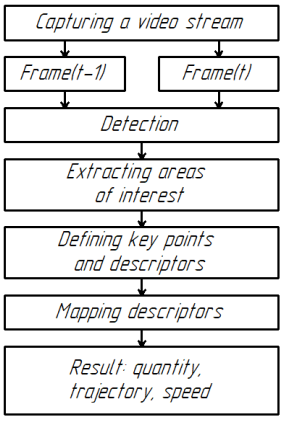


**Fig. 1.** An algorithm for estimating the flow of vehicles

# 4 Results and Discussions

## 4.1 The result of the evaluation of deep neural networks for vehicle detection

This section presents the results of vehicle detection experiments. The analysis of each of these experiments includes many measurements such as accuracy, number of parameters, floating point operations (FLOP), memory consumption and processing time.

To measure the effectiveness of a vehicle detector, indicators such as the measure of intersection of predicted and true boundaries containing the vehicle (Intersection, I), completeness (Recall, R) and accuracy (Precision, P) of object detection are used. they are used.

The intersection of the predicted and reference bounding box $I(1)$ indicates how well the coordinates of the bounding box were predicted by the full-convolutional neural network compared to the true markup.

$$I = \frac{S_I}{S_f + S_{gt} - S_I} \tag{1}$$

Where $S_I$ is the area of intersection of the predicted and true bounding boxes, $S_f$ is the area of the predicted bounding box, $S_{gt}$ is the area of the true bounding box.

The recall coefficient $R(2)$ indicates the sensitivity of the algorithm to type 2 errors, i.e. gaps, and is equal to the ratio in the true markup of the number of correctly predicted elements to this number.

$$R = \frac{tp}{tp + fn} \tag{2}$$

Where $t_p$ are the true-positive objects that we expected to see and received at the output, $f_n$ are the false-negative objects that we expected to see, but the algorithm did not determine them.

The accuracy of $P(3)$ shows that the sensitivity of the algorithm to errors of the first type is false positive and is equal to the proportion of the number correctly predicted in the bounding rectangles predicted by the algorithm.

$$P = \frac{tp}{tp + fp} \tag{3}$$

Where $f_p$ are false–positive objects that should not have been at the output, but the algorithm returned them by mistake at the output.

Detailed results of accuracy, completeness, and intersection measures are presented in table 2. Table 3 shows a list of models sorted by accuracy with characteristics such as FPS, memory, number of operations per second and the number of parameters of each model.

**Table 2.** Vehicle detection accuracy results obtained using each model.

| Models | Intersection (I), % | Precision (P), % | Recall (R), % |
|---|---|---|---|
| Faster R-CNN Inception V2 | 85,24 | 80,15 | 93,88 |
| Faster R-CNN Inception Resnet V2 | 80,43 | 79,35 | 81,63 |
| Faster R-CNN Resnet 101 | 90,38 | 78,73 | 93,88 |
| SSD Mobilenet V2 | 81,25 | 75,48 | 60,41 |
| SSD Inception V2 | 79,11 | 67,34 | 58,03 |

**Table 3.** Characteristics of models for vehicle detection.

| Models | Precision (P), % | FPS, 1/c | Memory, MB | FLOPS, ($*10^9$) | Number of parameters ($*10^6$) |
|---|---|---|---|---|---|
| Faster R-CNN Inception V2 | 80,15 | 9,09 | 2175,21 | 120,62 | 12,89 |
| Faster R-CNN Inception Resnet V2 | 79,35 | 1,54 | 18250,45 | 1837,54 | 59,41 |
| Faster R-CNN Resnet 101 | 78,73 | 5,13 | 6134,71 | 625,78 | 62,38 |
| SSD Mobilenet V2 | 75,48 | 30,30 | 282,5 | 6,4 | 15 |
| SSD Inception V2 | 67,34 | 25 | 284,51 | 7,59 | 13,47 |

Analyzing the data from table 3, we can say that the use of denser blocks in neural networks with the ResNet architecture leads to an increase in FLOPS and calculation time for the Faster R-CNN detector. It is worth emphasizing that the Mobilenet-V2 SSD is the model with the least number of FLOPS and the highest processing speed.

### 4.2 The results of the algorithm for estimating the flow of vehicles

Based on the results of the evaluation of deep neural networks, the Mobilenet-V2 neural network model was selected for vehicle detection, the result of which is shown in Figure 2.

To visualize the operation of the descriptor matching step, the key points of the most similar descriptors were constructed (Figure 3).

Based on the comparison of key points and descriptors of each vehicle, the algorithm compares the vehicles in the frame at time *t* with all vehicles at time *t-1*, and assigns unique identification numbers from *frame(t-1)* to *frame(t)*. (Figure 4).

Based on these data, we can estimate the instantaneous speed, the average in a given area, as well as the trajectory of each vehicle.

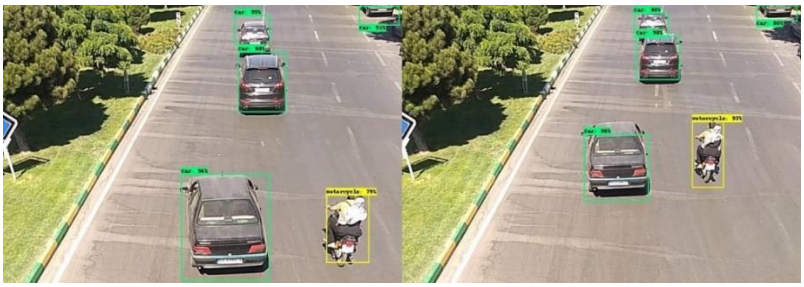

**Fig. 2.** Vehicle detection

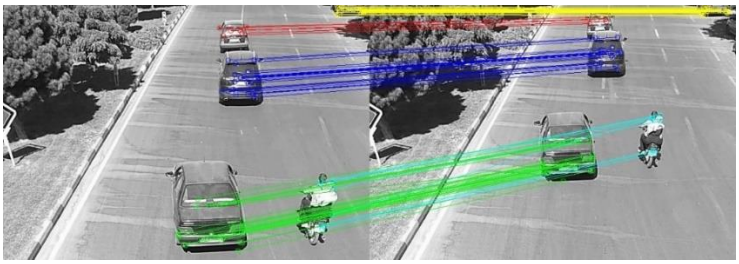

**Fig. 3.** Mapping descriptors

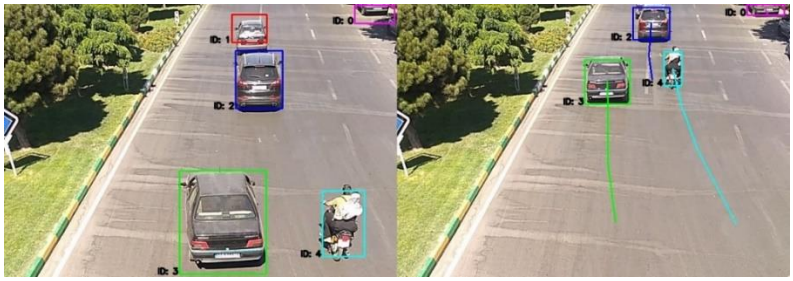

**Fig. 4.** Assignment of an identification number

## 5 Conclusion

As a result of work on this study, experimental comparisons of five neural network models for detecting vehicles were carried out, an algorithm for estimating traffic flow based on the SIFT method was developed, key characteristics of neural network models were analyzed: accuracy, speed, memory consumption, the number of FLOPS and the number of convolutional neural network parameters, and the results of the algorithm were presented.

It has been revealed that the neural network models Faster R-CNN Inception V2, Faster R-CNN Inception Resnet V2, Faster R-CNN Resnet 101 have relatively high accuracy, however, they greatly lose in detection speed, which makes their use impossible when detecting in real time. It should be emphasized that SSD Mobilenet V2 has the highest processing speed compared to other models, which is the fastest model of all detectors and exceeds the accuracy of SSD Inception V2.

Based on the results of the algorithm, it was shown that the SIFT method copes well with the task of comparing vehicle descriptors, which allows you to assign unique identification numbers on a sequence of frames for accurate counting of vehicles, their trajectories and speeds.

This work is the basis for further research on predicting congestion based on a recurrent neural network.

## References

1. Y. Du, X. Liu, Y. Yi, K. Wei, Sensors, **23**, 8844 (2023) doi.org/10.3390/s23218844
2. R. Rajamoorthy, G. Arunachalam, P. Kasinathan, R.Devendiran, P. Ahmadi, S. Pandiyan, S. Muthusamy, H. Panchal, H.A. Kazem, P. Sharma, Energy Sources Part A Recovery Util. Environ. Eff, **44**, 3555–3575 (2023) doi.org/10.1080/15567036.2022.2067268
3. B. Yu, H. Zhang, W. Li, C. Qian, B. Li, C. Wu, Sensors, **21**, 7118 (2021) https://doi.org/10.3390/s21217118
4. G. Wieczorek, S.B.u.d. Tahir, I. Akhter, J. Kurek, Sensors, **23**, 1731 (2023) doi.org/10.3390/s23031731
5. Y. Miao, F. Liu, T. Hou, L. Liu, Y. Liu, Chinese Automation Congress, **1**, 6617-6621 (2020) doi:10.1109/CAC51589.2020.9326819
6. A.T. Tajar, A. Ramazani, M. Mansoorizadeh, Journal of Real-Time Image Processing, **18**, 2389–2401 (2021) doi:10.1007/s11554-021-01131-w
7. E. Bochinski, V. Eiselein, and T. Sikora, IEEE International Conference on Advanced Video and Signal Based Surveillance (2017) doi:10.1109/AVSS.2017.8078516

8. Peng Liu, Guoyu Wang, Zhibin Yu, Xinchang Guo, Weigang Lu, Computers & Electrical Engineering, **78**, 22-31 (2019) doi.org/10.1016/j.compeleceng.2019.06.019.
9. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, (2015).
10. S. Ren, K. He, R. Girshick, J. Sun, Neural Information Processing Systems, **39**, 1137-1149 (2015).
11. W. Liu, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, European Conference on Computer Vision, **1**, 21–37 (2016) doi:10.1007/978-3-319-46448-0_2.
12. K. He, X. Zhang, S. Ren, J. Sun, IEEE conference on computer vision and pattern recognition, **1**, 770–778 (2016), doi:10.1109/CVPR.2016.90.
13. S. Ioffe, C. Szegedy, Proceedings of Machine Learning Research, 37, 448-456 (2015), cite arXiv: 1502.03167.
14. C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, AAAI Conference on Artificial Intelligence (2017), cite arXiv:1602.07261.
15. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017), cite arXiv:1704.04861.
16. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, IEEE Conference on Computer Vision and Pattern Recognition, **1**, 7310-7319 (2017) doi:10.1109/CVPR.2017.351.
17. Álvaro Arcos-García, Juan A. Álvarez-García, Luis M. Soria-Morillo, Neurocomputing, **316**, 332-344 (2018) doi.org/10.1016/j.neucom.2018.08.009.
18. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft Coco: Common Objects in Context, 740–755 (2014) doi.org/10.48550/arXiv.1405.0312
19. K. Mu, F. Hui, X. Zhao, Multiple The Journal of Information Processing Systems. **12**, 183–195 (2016**)** doi:10.3745/JIPS.02.0040