

Real-Time Vehicle Type Detection and Counting from Road Camera Video

Denis Zuenko^{a,b}, Ilya Makarov^a

^aNational Research University, Higher School of Economics, Moscow, Russia

^bSkolkovo Institute of Science and Technology (Skoltech), Moscow, Russia

Abstract

In this paper, we study automatic recognition and counting of vehicles in the wild. For this problem, we tested several object detection models for car type recognition among five classes: Bicycle, Bus, Car, Motorcycle, Truck, Van. We extend existing dataset in order to balance classes and achieve classification quality for detected cars with 92% mAP.

Keywords

vehicle detection, computer vision, object detection, road scene understanding

1. Introduction

Fast traffic detection and counting is a general approach for planning and analytic in traffic management intelligent systems. In many cases, the desire is to increase the level of details by additional categorization of vehicle types and make it fast enough to process a lot of cameras around the city. This allows a more robust analysis and profiling of users of the transportation infrastructure. This is necessary for assessing the effects of possible modifications in the existing controlling system.

There are exist several of different types of techniques in which can be used sensors (e.g., the MAVE-L product line of AVE GmbH¹, traffic counters from SICK GmbH²), cameras to recognize vehicles. In case of sensors require laborious installation and a significant amount of costly hardware, however, cameras are widely used for speed detection, lane intersection detection etc. (e.g., in Moscow around 200,000). The benefit is obvious: installing additional a framework or updating it to an existing solution in a large city will be cheaper. That's we choose as an objective task object detection.

However, the reliability of camera-based technologies may depend on environmental factors, such as poor lighting, weather conditions, or changes in viewing angle, and therefore the system developed must be resistant to such changes. As well as we need an accurately annotated balanced sets of images or video, which is another challenge.

We observed the most recent state-of-the-art object detection methods. You only look once (YOLO) [1] is an object detection system targeted for real-time processing and has a quite balanced trade-off between accuracy and speed, which allows us to process images in real-time. As was mentioned before, we used the Single Shot Detector (SSD) [2] based on MobileNetv2 with default parameters and with

MACSPro'20: Modeling and Analysis of Complex Systems and Processes, October 22–24, 2020, Venice, Italy


EMAIL: neron.v2@gmail.com (D. Zuenko); iamakarov@hse.ru, corresponding author (I. Makarov)

URL: <http://hse.ru/en/staff/iamakarov> (I. Makarov)

ORCID: 0000-0002-3308-8825 (I. Makarov)

© 2020 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<http://www.ave-web.de/>

²<https://www.sick.com/de/de/>

replaced Conv1x1 to the dimensional reduction layer as were proposed in Adaptive Mixture of Low-Rank Factorizations for Compact Neural Modeling (ALRF) [3], which compressed the whole network by 40%, but made it more sensitive to the angle of the camera. In further work, we want to compress models to make it even smaller and faster, because Nvidia closed commercial use of their Geforce GPUs and now if you should buy a brand new Tesla, so we need to compressed network.

Vehicle Classification is done differently with object detection because we don't have boxes annotation for parsed images³. We used deeper classes: typical car, firetruck, taxi, police, ambulance. This approach provides essential information about the area of the camera and can help control traffic depending on these special classes.

Vehicle counting is carried out using the virtual line method. This virtual line acts as a counter from which the count is updated. For each vehicle that enters into the frame and crosses the virtual line, the count is incremented. The classification and counting results are shown below, along with their counts. So the pipeline is obvious, we detect a car, then classify it, then count intersection.

2. Related works

Following the idea proposed by [4], the potential solutions for crowd scenes analytic can be classified into three categories: detection-based methods, regression-based methods, and density estimation-based methods. By combining deep learning, the CNN-based solutions show strong ability in this task and outperform the traditional methods. Detection-based method's pipeline is comprising foreground extraction, target localization, tracking, and trajectory classification. Regression-based methods are aiming at learning a regression function using features of detection regions and exploit that for counting. Density estimation-based methods track a set of features of target objects and cluster their trajectories for counting them.

For object detection task there are exist a lot of CNN based approaches, that can give you state-of-the-art accuracy as well as performance. It has a lot of benefits against author approaches because we can extract a lot more information from that. However, their performance depends on the scenarios where they are used. In [5] one of the major challenges to using aerial images to accurately detect cars and count them in real-time for traffic monitoring purposes. Authors showed that YOLO generalizes pretty good (quality 98.81%) even in that camera angle. And it took them 0.057 ms for one image with Geforce GTX 1080.

For the classification task, in [6] they compare several methods for classification different car brand, type, name and reached 87.6% accuracy on the Stanford Cars-196 dataset. Another classification and car recognition method [7] showed that if we detect with SSD then classify using two methods (Resnet in their work) it shows 98,36 % accuracy against SSD to classify object (90% accuracy), even using huge VGG[8] architecture within.

3. Dataset

The main goal is to build a dataset that it can give us a better generalization of model, higher balanced classes. Because in most common dataset class - "car" means all cars. We collected:

- Images from fans web-site⁴, this data provide us around 180,000 images of police and ambulance.

³<http://avto-nomer.ru/>

⁴<http://www.policecarwebsite.net/>

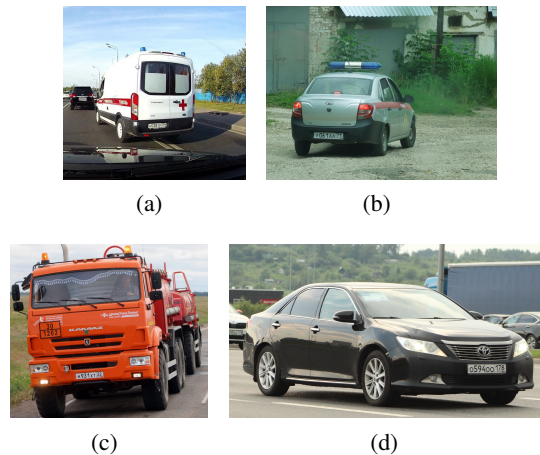


Figure 1: Examples: (a) ambulance; (b) police; (c) special services; and, (d) typical car.

- Videos from audio dataset⁵, which provided us a firetruck and ambulance images.
- The biggest one we claimed from number plate web-site⁶. The classes distribution:
 - Police car 37,603
 - Typical car 131,225
 - Taxi 41,326
 - Firetruck 9,263
 - Ambulance 22,020
 - Other special services 33,577
 - Vehicle's brand and model name as addition feature.

Clearing, re-downloading, optimizing, annotating still in process, because we have a lot of images. But this data is still has lack of night or evening pictures. Examples are shown in Figure 1.

For testing, we used video from Belgorod city. Which are not annotated for counting and detection task, but gives us understanding how our algorithm performs in the wild because of consist of 3 different types of weather, parts of a day, angles.

We want to mention that we do not provide any counting values nor train, validation and test splits because of annotation still in process. But if we get something new, we split it by 70-20-10 % for each category.

4. Methods

4.1. Classification

After collecting data we started to annotate it in a semi-supervised manner using DenseNet[9], which seemed pretty balanced for the size and robustness. To achieve better accuracy we used pre-trained imagenet model. The results are shown in Figure 2.

⁵<http://tiny.cc/3ucx8y>

⁶<http://avto-nomer.ru/>

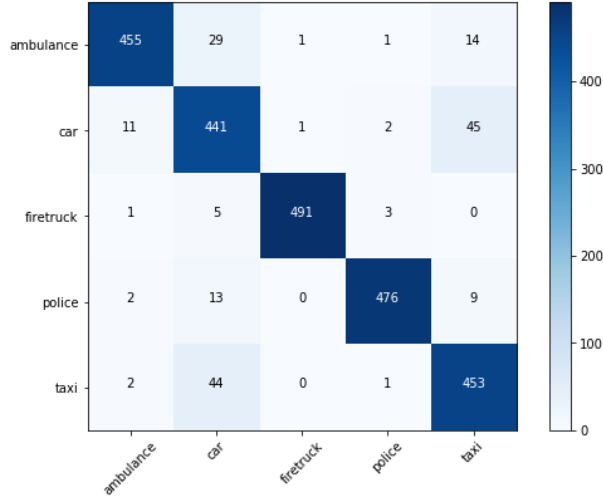


Figure 2: Confusion matrix of testing part of our data with DenseNet, mAP - 92%

4.2. Object detection

In the first we started to train the MobileNetv2[10] based SSD network with OpenImagesDatasetV2, image-size 512x512. This gave us an understanding of where to go next and whether a network trained on a common angle will behave on a specific one. Fig. 3-(b). In general, this algorithm showed a good result, despite the fact that class confidence was low. But we consider the specific position of the camera in that result.

Then we started to optimize SSD network by replacing Conv1x1 layers to the special layer in which, authors propose to use an unnormalized learned mixture of low-rank factorizations with mixing weights calculated adoptive based on the input. More specifically, denoting the input by h and the number of mixture components by K and decompose a large weight matrix by:

$$W(h) = \sum_{k=1}^K \pi_k(h) U^{(k)} (V^{(k)})^T$$

where $\pi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^K$ is the function which maps each input to its mixture coefficients, an $U^{(k)} \in \mathbb{R}^{m \times d/K}$, $V^{(k)} \in \mathbb{R}^{n \times d/K}$. This is evidenced by rewriting $W = [\pi_1 U^{(1)}, \dots, \pi_K U^{(K)}] [V^{(1)}, \dots, V^{(K)}]^T$.

In our case, π is the sigmoid function, but it can be a small neural network. However, it leads us to the limitation of this method, so we have three parameters in each new layer (at least 27 of them): the rank of decomposition, number of pairs $(U^{(k)}, V^{(k)})$, size of π function hidden layer. And that leads us to the main limitation of these methods if we compare it to other methods. In Fig. 3-(c), we showed the result of it, unfortunately, to make it show at least something, we low our thresholds to the lowest value. Seems that it over-fitted to the OpenImagesDataset.

At this stage of work, YOLOv3[1] was chosen as our final algorithm because it has few incremental improvements on YOLOv2. For example, a better feature extractor, DarkNet-53 with shortcut connections as well as a better object detector with feature map upsampling and concatenation. And it is published as a 2018 arXiv technical report with more than 200 citations. It was widely tested and showed robustness on various computer vision tasks, benchmarks and datasets. In Fig. 3-(a) we showed the result of it. With image-size 512x512 it can detect small objects in real time (36 fps on my Geforce GTX 1060).

4.3. Object counting

The existing video-based vehicle counting systems usually apply various algorithms to count vehicles, such as setting baselines [11]. Based on virtual line detection is suitable for counting vehicles with high speed. In traffic congestion, the vehicles are close to each other and move at a low speed; thus, there is a greater risk of counting two adjacent vehicles as one. That why tracking is the necessary part of the Algorithm 1.

Algorithm 1: Vehicle counting

Result: dictionary of objects and their counts

Input: frame, line or mask, detector algorithm;

while *frame is not None* **do**

 Update all the existing vehicles;

if *any of the matches fits this vehicle* **then**

 | Nearest point to the vehicle (tracking);

else

 | Update value that we haven't seen the vehicle in recent frame.

end

 Add new vehicles based on the remaining matches;

 Count any uncounted vehicles that are past the divider and check frame seen vehicle;

 Remove vehicles that have not been seen long enough;

if *max unseen frames \leq frames since seen* **then**

 | remove vehicle from base;

else

 | continue;

end

 print info;

 Stream it to the web-page;

end

Where things become difficult:

1. There are more than one lanes per direction. Therefore, it is quite possible that two cars fit through the region of interest at the same time thus creating just one path. Even worse, a car can be masked by a bus (be behind it) and never be counted. The only way to counteract this is to repeat the same ultra-simple "counting" process at multiple points, but it not worth the computation speed.
2. The virtual line can be intersected in different places, angles etc.
3. Its hard to understand where we should put the line or mask points.
4. Finding and interpolating vehicle's points are not effective, better to use smart tracking.
5. The counting accuracy in the daytime scenario is higher than that in the nighttime scenario, which coincides with the expectation. The uniformity of illumination in the daytime is higher than that in the nighttime. Thus, the risk of a vehicle being blended into the background at night is much higher than in the day.
6. Small objects (our tracking limit).
7. Make it efficient for the classification (in the process).

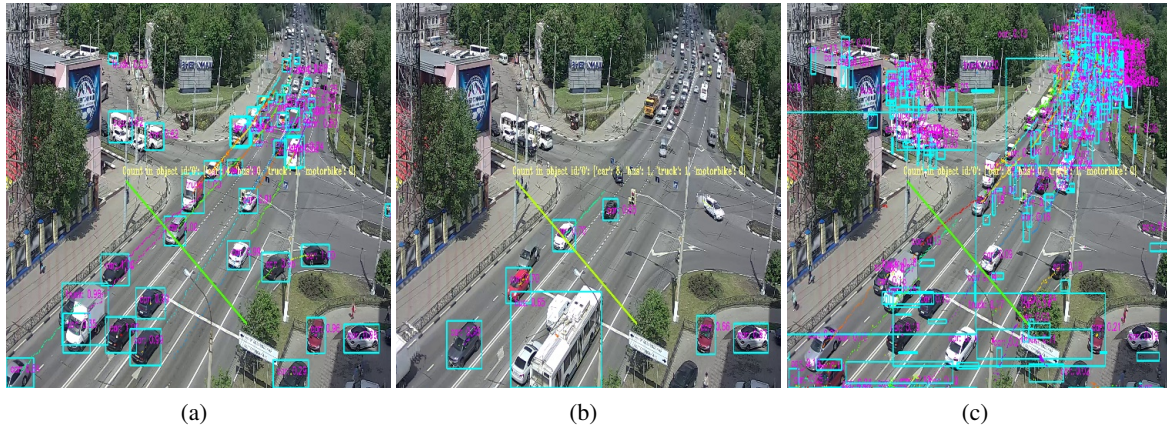


Figure 3: Results of object detection: (a) pre-trained YOLOv3; (b) trained SSD; and, (c) trained SSD with ALRF;

5. Results and Analysis

In our work, we tried to apply the easiest, fastest, and most accurate models in the field of object detection. But the most accurate and accurate result showed us YOLOv3 as evidenced by his fast on the pre-trained 'fast' model. We also showed that although ALRF shows a serious degree of compression for the popular MobileNetv2 architecture for small datasets, but for more complex tasks, it requires a smarter approach to selecting hyper-parameters.

When we trained the classification faced the limited angle of the objects, as well as for detection, and as a consequence of the generalization of the models, so we decided to collect own dataset and annotate it in order to improve the quality of the model and the final validation of the work.

For the counting task, in general, the algorithm shows itself very well, although there are some inconveniences connected with the limited models and the lack of a GUI for marking virtual lines and a mask. Examples of our system in work are shown in Figure 3.

6. Conclusions and Future Work

Information about traffic flows is important in modern cities. They can be used for vehicle detection, counting and monitoring tasks. Based on state-of-the-art sequential image processing technologies we showed a good quality at specific corners of the camera. But in order to increase the quality, we began to collect and annotate our VehiclesIntheWild (VIW) dataset. It will cover the following tasks of the brand and types of cars, their number in a certain area on the video. We also plan to integrate smart tracking into our counting algorithm and make it more robust, as well as develop a convenient interface for marking lines or areas. The biggest challenge awaits us in the future is the nightly data. But in recent times, many interesting architectures have been shown for generating data using the GANs (Generative adversarial network).

References

- [1] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

- [2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, Springer, 2016, pp. 21–37.
- [3] T. Chen, J. Lin, T. Lin, S. Han, C. Wang, D. Zhou, Adaptive mixture of low-rank factorizations for compact neural modeling, <https://openreview.net/forum?id=BlEHgu-Fim> (2018).
- [4] C. C. Loy, K. Chen, S. Gong, T. Xiang, Crowd counting and profiling: Methodology and evaluation, in: Modeling, simulation and visual analysis of crowds, Springer, 2013, pp. 347–382.
- [5] B. Benjdira, T. Khurshed, A. Koubaa, A. Ammar, K. Ouni, Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3, in: 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), IEEE, 2019, pp. 1–6.
- [6] K. Valev, A. Schumann, L. Sommer, J. Beyerer, A systematic evaluation of recent deep learning architectures for fine-grained vehicle classification, in: Pattern Recognition and Tracking XXIX, volume 10649, International Society for Optics and Photonics, 2018, p. 1064902.
- [7] B. Satar, A. E. Dirik, Deep learning based vehicle make-model classification, in: International Conference on Artificial Neural Networks, Springer, 2018, pp. 544–553.
- [8] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
- [9] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
- [10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.
- [11] T.-H. Chen, Y.-F. Lin, T.-Y. Chen, Intelligent vehicle counting method based on blob analysis in traffic surveillance, in: Second International Conference on Innovative Computing, Information and Control (ICICIC 2007), IEEE, 2007, pp. 238–238.