Physics and Chemistry of the Earth 110 (2019) 176–184

Contents lists available at ScienceDirect

Physics and Chemistry of the Earth

journal homepage: www.elsevier.com/locate/pce

An automatic traffic density estimation using Single Shot Detection (SSD) and MobileNet-SSD

Debojit Biswas[a], Hongbo Su[a,*], Chengyi Wang[b], Aleksandar Stevanovic[a], Weimin Wang[c]

[a] Department of Civil, Environmental and Geomatics Engineering, Florida Atlantic University, 777 Glades Rd, Boca Raton, FL, 33431, United States
[b] Lab of Remote Sensing Image Processing, Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, No. 20 Rd Datun, Chaoyang District, Beijing, 100101, China
[c] Shenzhen Environmental Monitoring Center, Shenzhen, China

A R T I C L E I N F O

Keywords:
Traffic density
Single shot detection
MobileNet
Convolution neural network
Feature maps
Visual Geometry Group

A B S T R A C T

Traffic density estimation is a very important component of an automated traffic monitoring system. Traffic density estimation can be used in a number of traffic applications – from congestion identification to macroscopic traffic control in urban environment. In this paper, we implemented Single Shot Detection (SSD) and MobileNet-SSD to estimate traffic density. SSD is capable of handling different shape, size and view angle of the objects. MobileNet-SSD is a cross-trained model from SSD to MobileNet architecture, which is faster than SSD. In this study, we show a key application area for the SSD and MobileNet-SSD framework. The advantages and shortcomings of the SSD and MobileNet-SSD framework were analyzed using fifty-nine individual traffic cameras. In addition, we compared the two algorithms with manually estimated density. The SSD framework shows significant potential in the field of traffic density estimation. SSD achieved 92.97% average detection accuracy in our experiment. On the other hand, the MobileNet-SSD achieved 79.30% average detection accuracy.

1. Introduction

Traffic congestion is a huge problem for the current world under a rapid urbanization. It's a global phenomenon across the world (Traffic Scorecard, 2018). For both developed countries like United States, Canada, China and developing countries like Bangladesh, India, Brazil, traffic congestion is a common problem for urban areas. Some traffic experts say the ultimate solution for this problem is a fully driverless car system. But with the current infrastructure there are two ways to reduce traffic congestion. One way is by building new alternative roads and expanding road network. The other way is by using new technologies to make the traffic system more efficient, for example, by using a more efficient urban traffic control or by better routing algorithms to utilize less crowded routes. Adopting both approaches may be necessary because in many scenarios, road expansion is not possible, and smart traffic signaling system or alternate route suggestion is not effective for extremely crowded events. This study is focused on estimating traffic density, which is an important element for these strategies to improve efficiency of urban traffic.

This study uses traffic cameras, which are already an integrated part of the traffic infrastructure. We considered the vibration effect caused by cameras being installed on a bridge or similar conditions. The Algorithm is trained with shaky and partially visible images, which will help to detect cars from low quality images. The SSD (Liu et al., 2016) and MobileNet-SSD (Howard et al., 2017) have been implemented for this study to detect cars. The light pole distance along the road is considered as a standard distance measurement to convert counted cars into density. Manual count and density estimation are performed to validate performance measurement.

We adopted the mature AI algorithms named SSD and Mobile-SSD. These AI algorithms have been optimized for image object detections and demonstrated to the AI algorithms with the highest efficiency (Liu et al., 2016). Most of the AI applications in transportation are focused on the auto pilot vehicle. The AI applications in traffic monitoring has insufficiently been conducted. As mentioned in Section 2, computer vision has been used for urban traffic monitoring, but the study to combine computer vision and AI for urban traffic monitoring is very limited. The study reported in this manuscript was built upon the prior study using traffic cameras to count vehicles on the road (Biswas et al., 2017).

* Corresponding author.
E-mail addresses: dbiswas2015@fau.edu (D. Biswas), suh@fau.edu (H. Su), wangcycastle@163.com (C. Wang), astevano@fau.edu (A. Stevanovic), towmwang@163.com (W. Wang).

https://doi.org/10.1016/j.pce.2018.12.001
Received 15 October 2018; Received in revised form 26 November 2018; Accepted 17 December 2018
Available online 18 December 2018
1474-7065/ © 2018 Elsevier Ltd. All rights reserved.

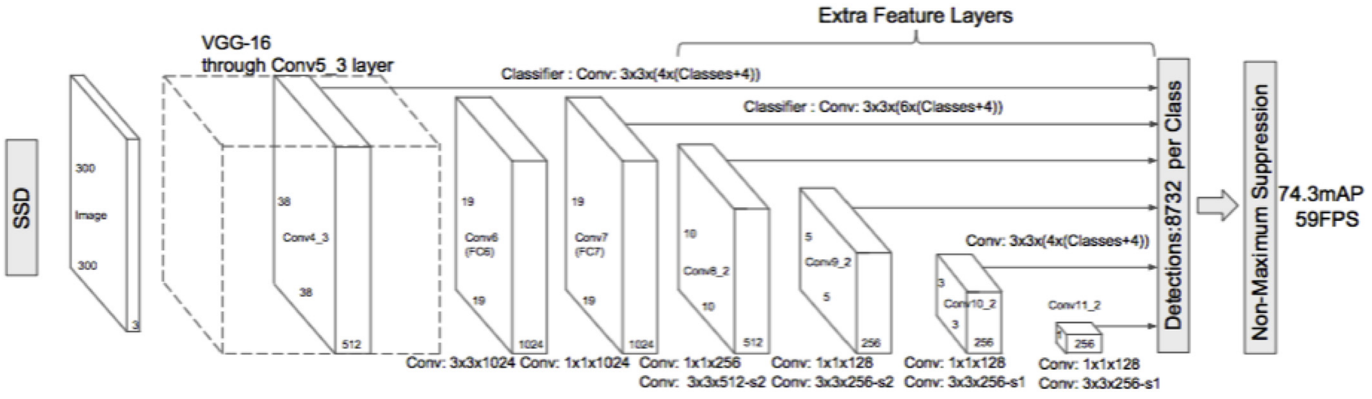


Fig. 1. SSD architecture (Liu et al., 2016).

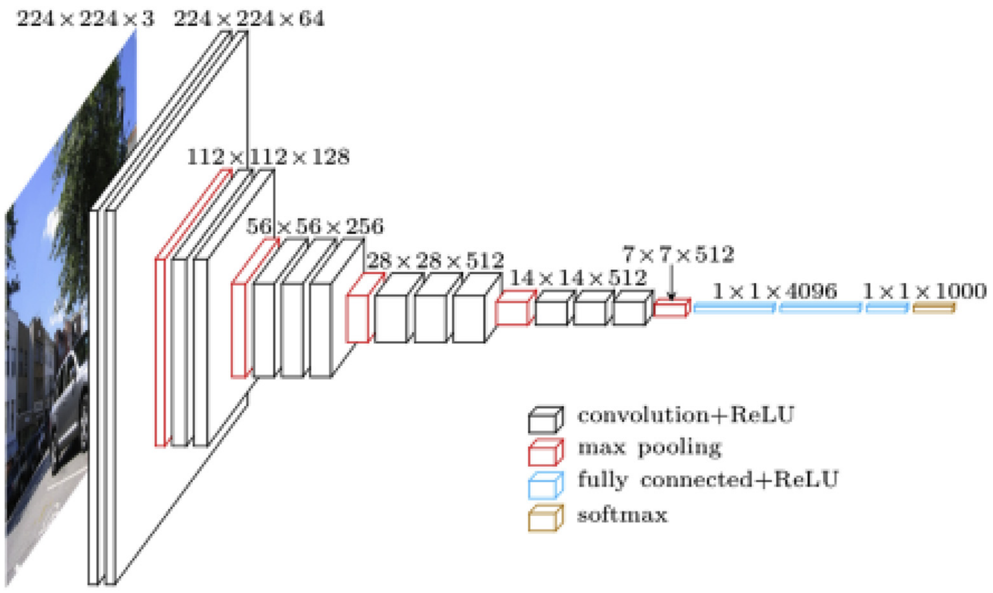


Fig. 2. A visualization of the VGG architecture (VGG architecture, 2018).

2. Related work

A lane wise car counting is a way to estimate the density with respect to time using traffic cameras was reported by Biswas et al. (2017). Magnetic loop detectors and surveillance cameras are widely used for traffic density estimation. Wireless vehicle sensors and speed guns are also an addition to these traditional techniques (Morarescu and Canudas-de, 2011; Tabibiazar and Basir, 2011) The hardwire specific technic like Magnetic loop detectors demands a lot of physical work. Also, it is very costly to install and maintain. That is why this type of technology cannot be adopted for large scale and real-time application.

Computer vision techniques are the future for urban traffic monitoring. An amazing survey is given at Buch et al. (2011) about the computer vision systems for urban traffic monitoring. Vehicle information is frequently composed from high pole cameras. Meng et al. (2011), explained three vision sensors to detect the number of vehicles. Tyagi et al. (2012), projected traffic density using cumulative road acoustics and the impact of noise was examined on the estimation. In (Yufei et al., 2012; Singh and Baibing, 2012; Van Hinsbergen et al., 2012), the authors used two wireless vehicle sensors located at both ends of the road segment to estimate vehicle density. Singh and Baibing (2012), solved a multiple road segment density estimation problem using Markov model approach. While Yufei et al., 2012, presented a Lagrangian state estimator-based approach and Van Hinsbergen et al. (2012), offered modified extended Kalman filter-based method for density estimation. Expectation Maximization approach is used to solve vehicle density estimation problem in (Ramezani et al., 2011, 2012). The average fraction of stop time of a probe vehicle is implemented to estimate vehicle density in (Artimy, 2007). Differentiate between the free and the congested traffic phases was a key purpose for density estimation in that paper. Uddin et al. (2015), proposed an area-based image processing technique for intelligent traffic light control system for the detection of traffic density. Anand et al. (2011), used a data fusion technique under heterogeneous traffic conditions to estimate density which is difficult to obtain from less disciplined traffic conditions.

From the literature review it is quite clear that the attempt to get the efficient traffic system is going on for decades. It is very important to update the traffic system with newly available technologies time to time. The existing car density estimation systems (Buch et al., 2011) either demand more hardware support or lack sophisticated algorithms Van Hinsbergen et al. (2012). This study uses existing traffic cameras with advance object detection algorithms (Liu et al., 2016; Howard et al., 2017) to get real time traffic density estimation.

This manuscript is ordered as follows. The implementation of SSD and MobileNet-SSD is descripted in section 3. Section 4 describes the datasets used for the experiment. The experimental results and an analysis of SSD and MobileNet-SSD are presented in Section 5. Section 5 also contains the discussion of the relationship between i) Training data, ii) Number of epochs and iii) Accuracy for SSD method. Finally,

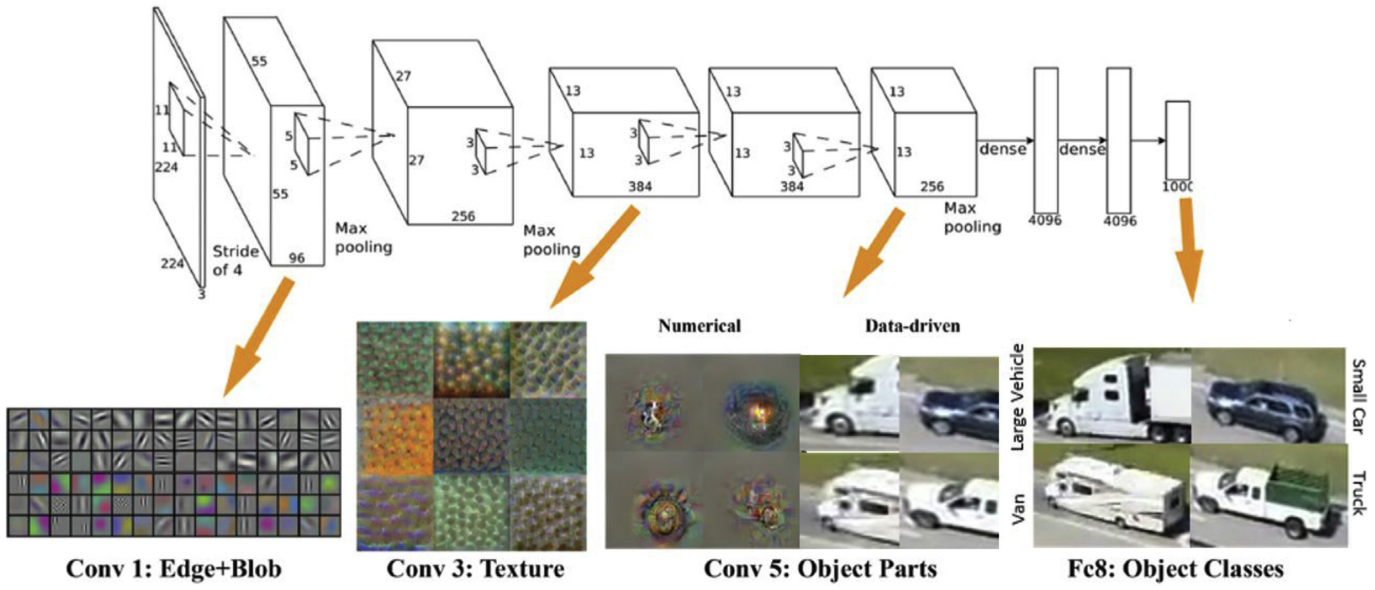


Fig. 3. VGG feature map visualization (VGG Feature Map Visualization, 2018).

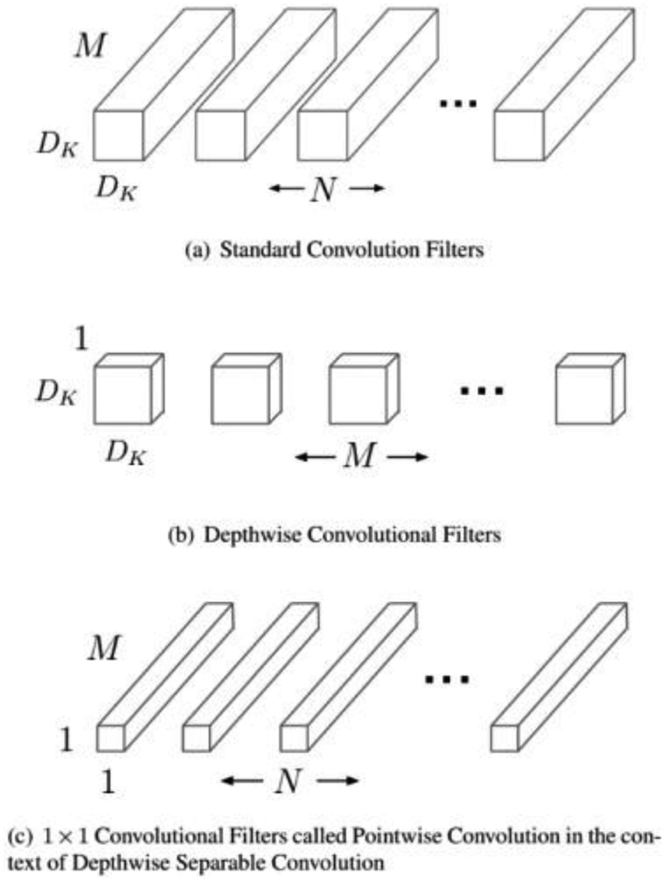

M
D_K
D_K
$\leftarrow N \rightarrow$
(a) Standard Convolution Filters
1
D_K
D_K
$\leftarrow M \rightarrow$
(b) Depthwise Convolutional Filters
M
1
1
$\leftarrow N \rightarrow$
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Fig. 4. The standard convolutional filters in Andrew et al., 2017

section 6 presented the conclusions.

3. Methodology

Methodology section describes the implementation of the algorithms used for the study. Python language is used to implement the algorithms.

3.1. Single Shot Detection (SSD)

The SSD is a feed-forward based convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. An auxiliary structure is added to the early network layers which are based on a standard architecture used for high quality image classification. The auxiliary structure is added to the network to produce detections with the following key features: a) Multi-scale feature maps for detection, b) Convolutional predictors for detection c) Default boxes and aspect ratios.

Fig. 1 illustrates the SSD architecture. We utilize the Visual Geometry Group (VGG)-16 layers up until conv_6 and then detach all other layers, including the fully-connected layers. The reason that the VGG-16 was used as the base network is because of its strong performance in high quality image classification tasks and its popularity for problems where transfer learning helps in improving results. A set of new CONV layers are then added to the architecture—these are the layers that make the SSD framework possible. As we can see from Fig. 1, each of these layers are CONV layers as well. This behavior implies that our network is fully-convolutional: we can accept an input image of arbitrary size — we are no longer restricted by the *224 X 224* input requirements of VGG. Fig. 2 explains the visualization of the VGG architecture.

There are two important components for SSD:

1. We progressively reduce the volume size in deeper layers, as we would with a standard CNN
2. Each of the CONV layers connects to the final detection layer.

The fact that each feature map connects to the final detection layer is important — it allows the network to detect and localize objects in images at varying scales. Furthermore, this scale localization happens in a forward pass. No resample of feature maps is required, enabling SSDs to operate in an entire feedforward manner—this fact is what makes SSDs so fast and efficient.

The SSD framework uses a modified version of MultiBox algorithm for bounding box proposals. The MultiBox algorithm starts with priors as described in Fig. 3. The priors are fixed size bounding boxes whose dimensions have been pre-computed based on the dimensions and locations of the ground-truth bounding boxes for each class in the dataset.

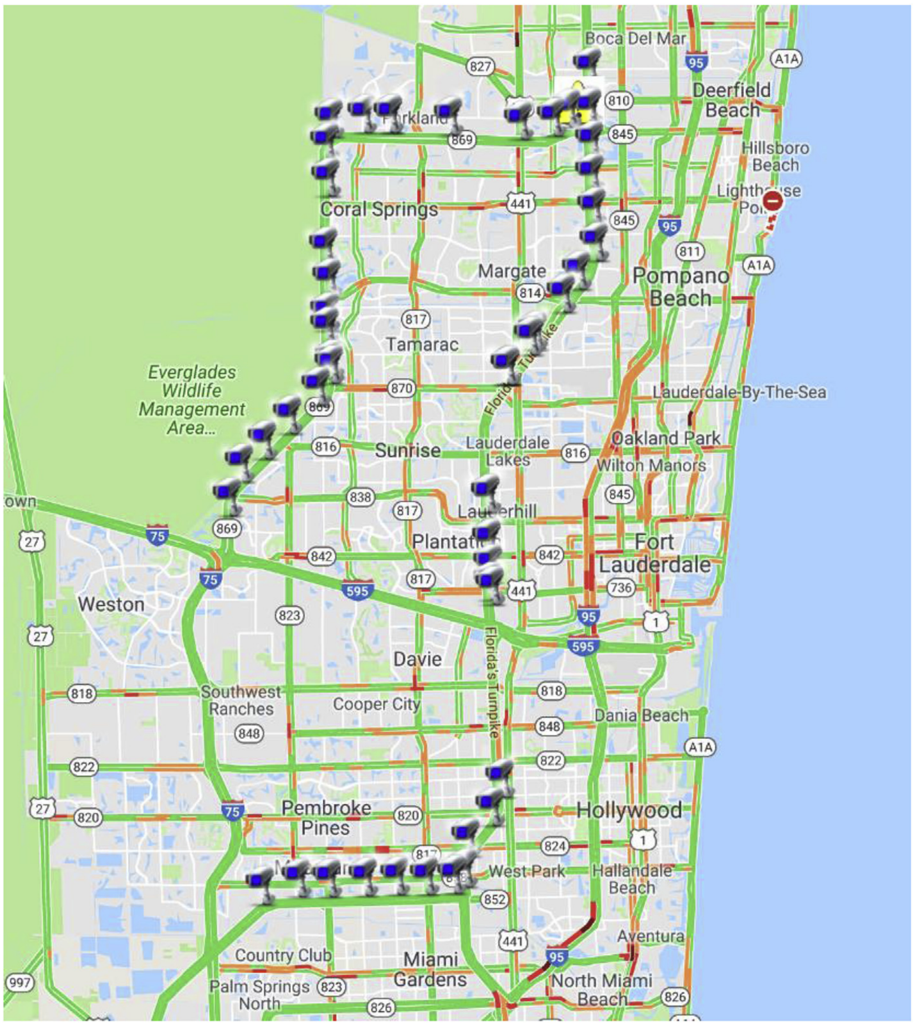


Fig. 5. Camera map on Florida's Turnpike.

We call these a “prior” as we're relying on Bayesian statistical inference, or more specifically, a prior probability distribution, of where object locations will appear in an image as described in Fig. 3. Different size of the boxes defines different type of features. SSD predicts scores of each of the classes for each of the feature maps that specify the presence of a class instance in each of those boxes. SSD creates different resolution feature maps to efficiently discretize the space of possible output shape of the boxes. Fig. 3 shows how the network “sees” a given image across its feature maps.

Implementation of SSD: The most tedious job for any object detection algorithm is to create training data sets. We labeled 41044 objects over 4006 images. All the label image creates a.xml file which contains the detail information (location, height and width) about the label objects. Before presenting the label dataset to SSD, the mapping is created about the location of the datasets. During the training process we set the learning rate to 0.0004. Later the learning rate has improved with respect to the iteration. After the training process, a.caffemodel is created which is used for object detection for testing. For our experiment we used upto CONV9_2 (Fig. 1) label of the SSD network.

3.2. MobileNet-SSD

The general trend has been to increase the depth and computation complexity of the network to increase the performance of a network. But some real-time applications, like self-driving car, augmented reality, real time object recognition, need a faster network.

The MobileNet model is based on depth wise and point wise convolution architecture. When a convolution operation is done by these two steps the computation complexity is reduced. Fig. 4 (a) shows the conventional convolutional filter, which is four dimensional. Fig. 4 (b) shows depth wise convolutional filter, which is three dimensional and Fig. 4 (c) shows point wise convolutional filter which is a vector operation. The combine steps of 4. (b) and (c) take less computational complexity than 4(a) alone.

The MobileNet architecture is performed for this experiment as described by Howard et al., 2017). The first layer of the architecture is fully convolution, apart from that the rest of the MobileNet structure is built on depthwise separable convolutions. The final fully connected layer has no nonlinearity and feeds into a softmax layer for classification. The rest of the layers are followed by a batchnorm (Ioffe and Szegedy, 2015) and ReLU nonlinearity. Stride convolution in the depthwise convolutions handled down sampling for the first layer as well as rest of the network. A final average pooling reduces the spatial resolution to 1, counting depthwise and pointwise convolutions as separate layers, before the fully connected layer.

Implementation of MobileNet-SSD: SSD generated.caffemodel is the base model for MobileNet-SSD. The MobileNet architecture (stored as MobileNet.prototxt) used to cross train the SSD's.caffemodel by the MobileNet architecture. The cross training process is more time consuming than a fresh training of SSD network. The newly generated.caffemodel is now used for object detection.

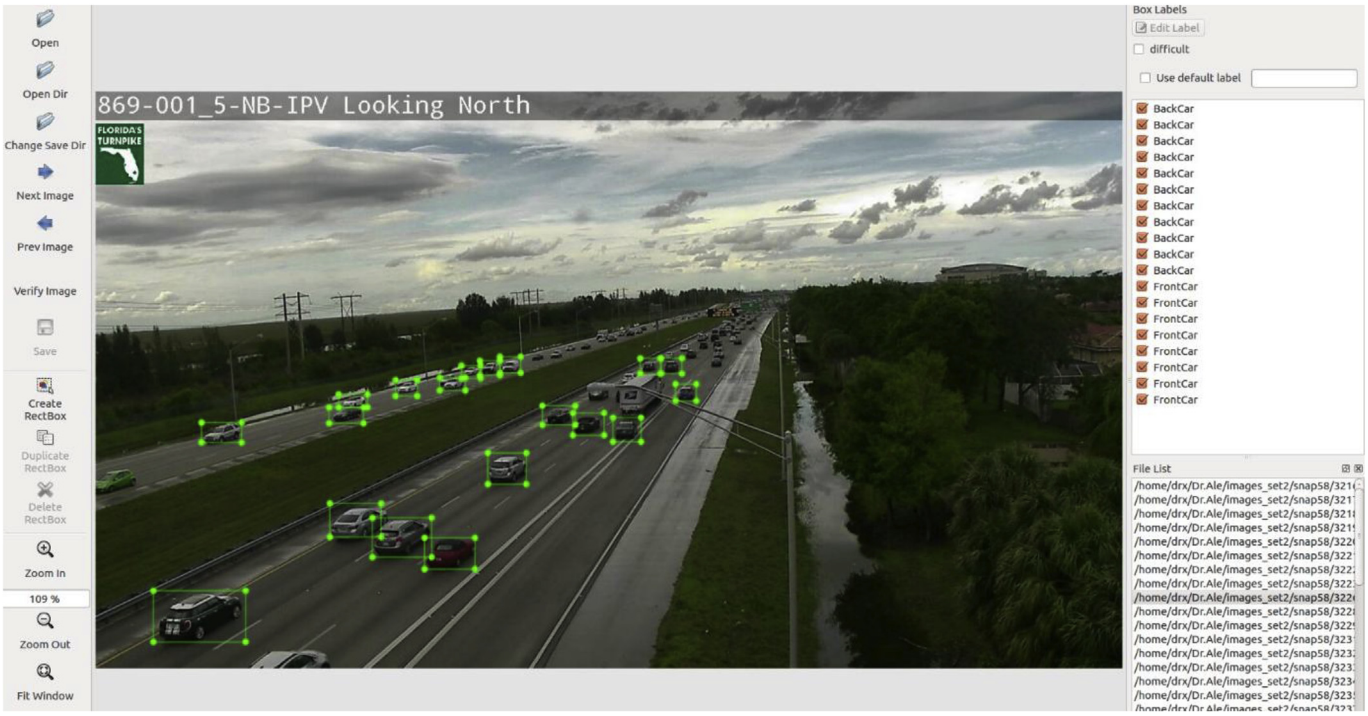


Fig. 6. Labeled vehicles shown in software ‘labelimg’.

Table 1
Number of labeled images for training.

Number of Images	Number of cars with Front Sides	Number of cars with Back Sides	Total Number of Cars
259	2184	2437	4621
465	3214	3613	6827
732	4975	5408	10383
985	7068	7534	14602
1245	8515	9062	17577
1485	9723	10349	20072
1738	10654	11359	22013
1981	11825	12563	24388
2223	13086	13897	26983
2510	14374	15310	29684
2793	15398	16407	31805
3086	16333	17278	33611
3319	17156	18089	35245
3557	18138	19103	37241
3817	19060	20190	39250
4006	19944	21100	41044

4. Datasets

Data is very important part of CNN based study, especially for training purpose. Fifty-seven cameras on Florida's Turnpike, one intersection camera for Sunrise/Andrews and one under bridge camera are used for this study. Fig. 5 shows the camera locations on Florida's Turnpike.

Images are collected form all the cameras using a script and labelling is done using an open source software “labeling” (LabelImg, 2018). The labelling screen is presented in Fig. 6.

For this study the cars are labeled as “Front” and “Back” car. As, the view from the traffic cameras cover both directions, it is appropriate to label the car as ‘Front’ or ‘Back’ to get more accurate density information direction-wise. We used small cars, pickup trucks, large trucks, sedans and SUVs for our training. Motorcycles are not used as a training data for this study. The training images are increased stepwise and training models (caffemodel) are created simultaneously, as one of the objectives for this study is to understand variation of accuracy with respect to the number of training images. Table 1 describes the number of “Front” and “Back” cars, which are labeled to create different sets of training data.

5. Result and discussion

SSD and MobileNet-SSD algorithms were tested on fifty-seven Florida's Turnpike expressway cameras, one intersection camera for Sunrise/Andrews and one under bridge camera. One hundred and five sample images were collected to test the algorithms. Manual counting was done to validate the proposed approach and estimate its accuracy. There were seven hundred and five “Front” cars and seven hundred and sixty-six “Back” cars manually counted to assess the accuracy of the algorithms. The result set for the algorithm is shown in Table 2, SSD and MobileNet-SSD, where the first column describes the model name which has the number of images used for the model and number of iterations. For an example ‘SSD259_10k’ means that 259 images were used, and 10,000 iterations were done to create the model. The average speed for training using two Nvidia GTX 1080 GPU and 128 GB RAM was 2k epoch (iterations)/hour. We used 10k iteration models for our experiments, that means the system takes 5hrs on an average for training. During the testing process models were created for every 5000 iterations. Like ‘SSD259_5k’, ‘SSD259_10k’, ‘SSD259_15k’ till “SSD259_50k’. Table 2 shows only the best model during the testing process. In most of the cases, models with either 10,000 or 15,000 iterations show the best results. From the SSD average accuracy column, it is evident that when fewer number of images are used, the accuracy was not good. Accuracy increased as the number of processed images increased. However, after certain level of images was reached (~3000 images) the accuracy was not improving significantly.

The MobileNet-SSD was cross trained by the SSD model. For example, ‘SSD259_10k_MN_50k’ indicates that ‘SSD259_10k’ was selected for cross training and 50,000 iterations were done with MobileNet architecture. The MobileNet-SSD started achieving more than 50% accuracy with ‘SSD1243_10k’ model. The previous models were giving less than 30% accuracy. For the MobileNet-SSD experiment, the best

Table 2
Detection accuracy assessment for SSD and MobileNet-SSD.

SSD Model Name	SSD		SSD Avg. Accuracy %	MobileNet-SSD Model Name	MobileNet-SSD		MobileNet- SSD Avg. Accuracy
	Front	Back			Front	Back	
SSD259_10k	209	221	29.24%	SSD259_10k_MN_50k	107	111	14.83%
SSD465_10k	237	201	29.92%	SSD465_10k_MN_50k	129	135	17.96%
SSD732_10k	260	222	32.93%	SSD732_10k_MN_50k	152	187	22.98%
SSD985_10k	268	304	38.85%	SSD985_10k_MN_50k	203	246	30.45%
SSD1243_10k	316	306	42.38%	SSD1243_10k_MN_50k	394	369	52.02%
SSD1485_10k	389	410	54.35%	SSD1485_10k_MN_50k	406	476	59.86%
SSD1738_10k	405	539	63.90%	SSD1738_10k MN_50k	460	453	62.19%
SSD1981_15k	456	564	69.15%	SSD1981_15k MN_50k	499	483	66.91%
SSD2223_15k	540	592	76.94%	SSD2223_15k MN_50k	510	529	70.70%
SSD2510_10k	608	656	84.95%	SSD2510_10k MN_50k	578	587	**79.30%**
SSD2793_10k	619	706	89.98%	SSD2793_10k MN_50k	554	551	75.25%
SSD3086_10k	690	632	90.18%	SSD3086_10k MN_50k	494	486	66.75%
SSD3319_10k	697	667	**92.97%**	SSD3319_10k MN_50k	488	489	66.52%
SSD3557_10k	658	647	88.89%	SSD3557_10k MN_50k	550	500	77.64%
SSD3817_10k	632	612	84.77%	SSD3817_10k MN_50k	343	304	44.16%
SSD4006_10k	637	634	86.56%	SSD4006_10k MN_50k	357	331	46.92%

Table 3
Density accuracy assessment for MobileNet-SSD.

Manual Count on 50 images for Camera Sunrise/ Andrews Intersection		Manual Density per image per 115 ft.		MobileNet-SSD Detection		MobileNet-SSD Detection		MobileNet-SSD Average Density Accuracy
Front	Back	Front	Back	Front	Back	Front	Back	
238	16	4.76	.32	222	14	4.44	.28	90.38%

model from SSD were chosen for every image sets and the models were cross-trained using MobileNet architecture. The reason behind choosing MobileNet-SSD is that it is faster than SSD. The speed achieved by the experiment using MobileNet-SSD is ~15–18 FPS, whereas SSD achieved ~5–7 FPS. The sacrifices that have to be made for improvement of the speed with MobileNet-SSD is a lower accuracy. It is observed during the testing process that MobileNet-SSD did not show good accuracy with far-view cameras. However, MobileNet-SSD performed decently with close-view cameras like Sunrise/Andrews Intersection camera.

As MobileNet-SSD performs well with larger objects, a close-view camera (Sunrise/Andrews Intersection) was chosen to estimate the density. Table 3 shows the density estimation result for the MobileNet-SSD algorithm. The distance on the road for this camera was 115 ft, which is shown between two red dots in the Sunrise/Andrews Intersection of Fig. 7. The distance was measured using 'Google map scale' for the intersection. MobileNet-SSD has achieved impressive density accuracy of 90.38%.

Table 4 shows the results for density estimation using the SSD algorithm. To test the algorithm, ten cameras were chosen. To convert the car count into the density, the adjacent pole distance is chosen. Florida turnpike has a standard distance between two adjacent "poles", based on which we used as a distance to calibrate the pixels in the image and converted the total number of cars within two adjacent poles to the car density. The distance was chosen by using the distances between lighting poles which is 220 ft on the Turnpike roads (check camera 3229 and 3232 from Fig. 8). When the poles are not available, we used the white and black line markers on the road. The white line is 10 feet long, the black line is 10 feet long, and the distance is 20 feet between each set. The Sunrise/Andrews camera does not contain poles or line markers, so distance is measured using google map scale between two points (red dots at Fig. 7). The distance measured was 115 ft. The average density accuracy achieved for the experiment is 82.90%.

Fig. 7 shows the result for the SSD algorithm which shows six turnpike sample detection results from cameras 3216, 3217, 3218, 3221 and 3223. All the cameras from Fig. 7 have different camera angles from each other. Camera 3216 has only front cars but far distance from the camera. Camera 3217 is blurry. But still SSD performs well in this situation. We have taken care of blurry vision frames during training time. Camera 3221 is also blurry, but the detection result is impressive. Camera 3218 has a reflective view. The detection result from SSD is still good for this scenario. Camera 3223 contains very far view small size cars, which is difficult to detect for most of the object detection algorithms. Under bridge camera is an ideal frame for object detection where object size is large and camera resolution is high (1280 × 720). Though the SSD algorithm performs well for most of the cameras, it does not perform well for Sunrise/Andrews Intersection. The reason behind this poor performance is fewer number of training data sets. Only thirty images were used for training process from this camera.

Fig. 8 shows the results for MobileNet-SSD algorithm. Detection results from two turnpike camera cameras 3229 and 3232 are presented here. Also, under the bridge and Sunrise/Andrews Intersection detection results are presented. MobileNet-SSD performs well for close view camera though few numbers of images were used for the training. However, the detection results from the camera on Turnpike, which are far-view camera, are not good.

6. Conclusions

This study opens a new path for traffic density estimation in real-time. We implemented two algorithms, SSD and MobileNet-SSD, by using the Python language. The SSD algorithm is slower (~5–7 FPS) but more accurate (92.97% (Table 2) detection accuracy) whereas the MobileNet-SSD is faster (~15–18 FPS) but less accurate (79.30% (Table 2) detection accuracy), with an exception that the close view camera achieved 90.38% (Table 3) density accuracy for MobileNet-SSD. These results suggest that, depending on various traffic and camera-view scenarios, we can choose different algorithms according to their potentials. Traffic density estimation can be done for the different times of the day to generate statistics, which can be helpful to optimize the performance of the traffic in urban networks.

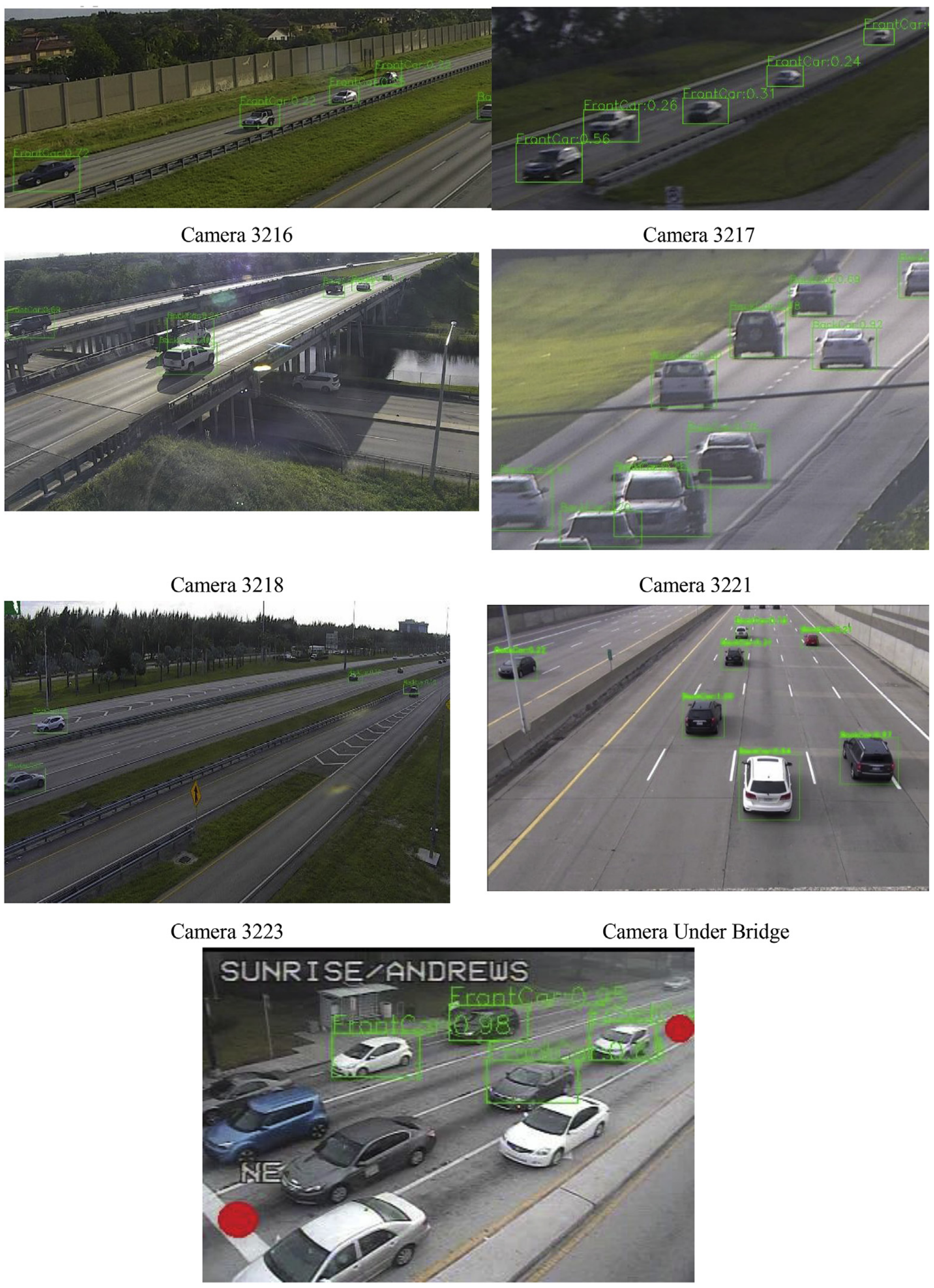


Fig. 7. Detection results for SSD.

Data availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of interest

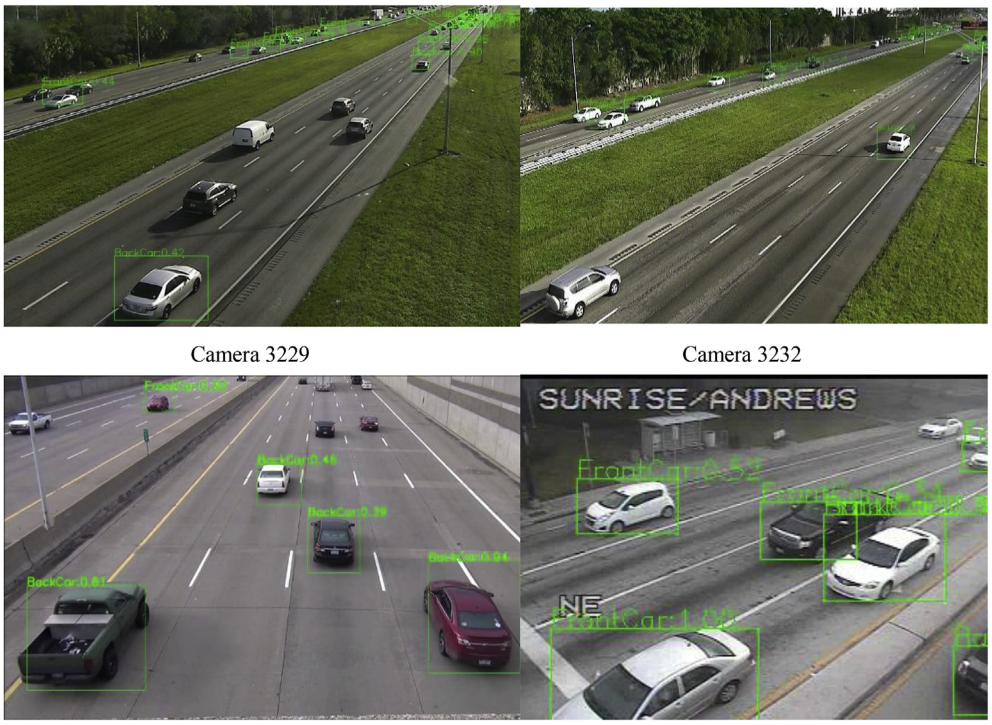
The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Funding statement

The work was supported by Guangxi innovation driven development special projects on “Product development and applications of nearshore dual frequency LiDAR detector” and a Florida Space Research Grant (No. NNX15_002). The authors are thankful to FDOT staff for their help with acquiring the data under the project BDV27 TWO 977–12 - “Development of a Traffic Map Evaluation Tool for Traffic Management Center Applications”. It should be noted that the opinions, findings and conclusions expressed in this publication are those of the authors and not necessarily those of the Florida Department of Transportation or the U.S. Department of Transportation.

Table 4
Density accuracy assessment for SSD.

Camera Name	Number of images	Manual		Manual Density per image per 440 ft.		SSD		SSD Density per image per 440 ft.		Average Density Accuracy%
		Front Count	Back Count	Front	Back	Front Count	Back Count	Front	Back	
3219	73	335	273	4.58	3.73	510	287	6.98	3.93	86.24%
3222	72	238	230	3.30	3.19	220	182	3.05	2.52	85.78%
3223	76	211	349	2.77	4.59	115	275	1.51	3.61	76.12%
3225	40	36	168	0.90	4.20	36	75	0.90	1.87	78.27%
3226	78	302	382	3.87	4.89	400	521	5.12	6.67	84.29%
3228	77	283	332	3.67	4.31	273	261	3.54	3.38	87.54%
3229	85	404	353	4.75	4.15	442	291	5.20	3.42	83.54%
3231	74	244	317	3.29	4.28	257	167	3.47	2.25	89.44%
3233	77	302	293	3.92	3.80	237	235	3.07	3.05	79.34%
3234	65	193	246	2.96	3.78	168	172	2.58	2.64	78.48%
Average Accuracy										**82.90%**



Camera Under Bridge Camera Sunrise/Andrews Intersection

Fig. 8. Detection results for MobileNet-SSD.

Author contributions

Hongbo Su conceived and designed the experiments; Debojit Biswas performed the experiments and analyzed the data; Debojit Biswas and Hongbo Su wrote the paper. Aleksandar Stevanovic, Chenyi Wang and Weimin Wang revised the paper.

References

Anand, R.A., Vanajakshi, L., Subramanian, S.C., 2011. Traffic density estimation under heterogeneous traffic conditions using data fusion. In: IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, June 5-9.

Artimy, M., 2007. Local density estimation and dynamic transmission-range assignment in vehicular ad hoc networks. IEEE Trans. Intell. Transport. Syst. 8 (3), 400–412.

Biswas, D., Su, H., Wang, C., Blankenship, J., Stevanovic, A., 2017. An automatic car counting system using OverFeat framework. Sensors 17, 1535.

Buch, N., Velastin, S.A., Orwell, J., 2011. A review of computer vision techniques for the analysis of urban traffic. IEEE Trans. Intell. Transport. Syst. 12 (3), 920–939.

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. 1704.04861.

Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 1502.03167.

LabelImg. (accessed on 5 March 2018), Available online: https://github.com/tzutalin/labelImg.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. 1512.02325.

Meng, C., Weihua, Z., Barth, M., 2011. Mobile traffic surveillance system for dynamic roadway and vehicle traffic data integration. In: 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 771–776.

Morarescu, I.C., Canudas-de, C.W., 2011. Highway traffic model-based density estimation. In: American Control Conference (ACC), pp. 2012–2017.

Ramezani, A., Moshiri, B., Abdulhai, B., Kian, A.R., 2011. Estimation of free flow speed and critical density in a segmented freeway using missing data and Monte Carlo-based expectation maximisation algorithm. IET Control Theory & Appl. 5 (1), 123–130.

Ramezani, A., Moshiri, B., Abdulhai, B., Kian, A.R., 2012. Distributed maximum likelihood estimation for flow and speed density prediction in distributed traffic detectors with Gaussian mixture model assumption. IET Intell. Transp. Syst. 6 (2), 215–222.

Singh, K., Baibing, L., 2012. Estimation of traffic densities for multilane roadways using a markov model approach. IEEE Trans. Ind. Electron. 59 (11), 4369–4376.

Tabibiazar, A., Basir, O., 2011. Kernel-based optimization for traffic density estimation.

In: IEEE Vehicular Technology Conference (VTC Fall), pp. 1–5.
Traffic Scorecard, 2018. INRIX. Available online: http://inrix.com/.
Tyagi, V., Kalyanaraman, S., Krishnapuram, R., 2012. Vehicular traffic density state estimation based on cumulative road acoustics. IEEE Trans. Intell. Transport. Syst. (99), 1–11.
Uddin, M.S., Das, A.K., Md Taleb, A., 2015. Real-time area based traffic density estimation by image processing for traffic signal control system: Bangladesh perspective. In: 2nd Int'l Conf. On Electrical Engineering and Information & Communication Technology (ICEEICT).
van Hinsbergen, C.P.I.J., Schreiter, T., Zuurbier, F.S., van Lint, J.W.C., van Zuylen, H.J., 2012. Localized extended kalman filter for scalable real-time traffic state estimation. IEEE Trans. Intell. Transport. Syst. 13 (1), 385–394.
VGG architecture. (accessed on 10 February 2018). Available online: https://www.cs.toronto.edu/~frossard/post/vgg16/.
VGG Feature Map Visualization. (accessed on 15 February 2018). Available online: http://vision03.csail.mit.edu/cnn_art/index.html.
Yufei, Y., van Lint, J.W.C., Wilson, R.E., van Wageningen-Kessels, F., Hoogendoorn, S.P., 2012. Real-time Lagrangian traffic state estimator for freeways. IEEE Trans. Intell. Transport. Syst. 13 (1), 59–70.