Efficient CityCam-to-Edge Cooperative Learning for Vehicle Counting in ITS

Honghui Xu, Zhipeng Cai, *Senior Member, IEEE*, Ruinian Li, and Wei Li, *Member, IEEE*

***Abstract*— Vehicle counting is a fundamental component in Intelligent Transportation System (ITS) for city traffic management. Although a number of vehicle counting approaches have been proposed, their essential drawbacks limit the efficacy of vehicle counting in real applications. In this paper, we propose a CityCam-to-Edge cooperative learning framework by cooperating multiple city cameras with an edge server to count vehicles more efficiently. Our learning framework consists of a lightweight feature extraction scheme deployed on the city cameras and a vehicle counting model implemented on the edge server. We devise the lightweight feature extraction scheme by leveraging multiple convolutional layers with few kernels in the design of deep learning architecture to reduce the utilization of parameters for feature extraction, so that the city cameras' memory consumption and the data transmission time can be greatly reduced. Moreover, we design two novel vehicle counting models, F2F-M and O2O-M, to improve the counting performance by exploiting the temporal correlation among videos captured from multiple city cameras in a frame-to-frame manner and a video-to-video manner, respectively. By combining the lightweight feature extraction scheme and the proposed vehicle counting models, we obtain two end-to-end vehicle counting models, Lite-F2F-M and Lite-O2O-M. Finally, via conducting extensive experiments, we demonstrate that Lite-F2F-M and Lite-O2O-M models outperform the state-of-the-art in terms of vehicle counting accuracy and time efficiency.**

***Index Terms*— CityCam-to-Edge, vehicle counting, cooperative learning, lightweight scheme.**

I. Introduction

THE U.S. national highway traffic safety administration (NHTSA) reported 32719 fatalities and 2.313 million injuries in 2013 [1]. Meanwhile, the U.S. federal highway administration (FHWA) predicts a 23% increase in vehicle miles traveled by 2032 (*i.e.*, 1.04% annual growth) [2]. With the increasing demand on surface transportation, its impact on traffic safety has become a major concern for transportation agencies [3]–[7]. To facilitate city traffic management, Intelligent Transportation System (ITS) [8]–[12] has been proposed by leveraging connected intelligent devices (*e.g.*, vehicle cameras, city cameras, *etc.*), aiming to alleviate the pressure of continued growing demand on transportation and to reduce roadways congestion [13]–[15]. Vehicle counting is one of the core components of ITS for city traffic management [16], in which the traffic flow information is obtained by counting vehicles. On the one hand, the real-time traffic flow information can be exploited for optimizing traffic signal light timing [17]–[19], gauging congestion level, and redirecting traffic along less congested routes [20]–[22]. On the other hand, the long-term traffic flow information can be employed by governments for replanning and redesigning infrastructure [23]–[25], making transportation systems more safe and efficient.

So far, a lot of research has been conducted to perform vehicle counting in various ways. One vein of research is to count vehicles based on magnetic sensors [26]–[31]. However, the installation and regular maintenance of these magnetic sensors usually cause inconvenience to people's daily life, such as lane closure and traffic disruption. Additionally, the reinstallation of these sensors is required when resurfacing or repairing roadways. What's worse, these sensors rely on pavement geometry, which means that a deterioration in pavement will result in unreliable data. Recently, vision-based models have been widely used to count vehicles more accurately without traffic disruption by dealing with city cameras' data. These vision-based models can be broadly classified into two categories. i) Conventional models, including the frame differencing methods [32], [33], the detection-based methods [34], [35], the tracking-based methods [36]–[39], and the density estimation-based methods [40], usually suffer the issues of low-frame-rate, high-occlusion (that means are some vehicles are hidden by other vehicles in traffic videos), and large-perspective (that means city cameras arranged along a road can capture traffic scenes with a large perspective) in traffic videos captured by city cameras. ii) As the state-of-the-arts, deep learning-based models [41]–[44] have been developed recently to improve counting performance, which are usually implemented on servers after receiving raw traffic videos from city cameras. However, it is so time-consuming to transmit the high-resolution traffic videos from city cameras to the servers in real applications. Therefore, it is challenging to design an efficient vehicle counting method to process high-resolution traffic videos while decreasing the data transmission time.

To solve this challenge, in this paper, we propose a CityCam-to-Edge cooperative learning framework for vehicle counting, in which a lightweight feature extraction scheme

Manuscript received 29 June 2021; revised 19 December 2021; accepted 26 January 2022. Date of publication 14 February 2022; date of current version 12 September 2022. This work was supported in part by U.S. National Science Foundation under Grant 1704287, Grant 1829674, Grant 1912753, and Grant 2011845. The Associate Editor for this article was W. Wei. *(Corresponding author: Wei Li.)*

Honghui Xu, Zhipeng Cai, and Wei Li are with the Department of Computer Science, Georgia State University, Atlanta, GA 30302 USA (e-mail: hxu16@student.gsu.edu; zcai@gsu.edu; wli28@gsu.edu).

Ruinian Li is with the Department of Computer Science, Bowling Green State University, Bowling Green, OH 43403 USA (e-mail: lir@bgsu.edu).

Digital Object Identifier 10.1109/TITS.2022.3149657

is designed for the city cameras, and deep learning-based vehicle counting models are developed for the edge server. The lightweight feature extraction scheme deployed on the city cameras can reduce the utilization of parameters by using multiple convolutional layers with few kernels in deep learning architecture for feature extraction, leading to a significant decrease in the city cameras' memory consumption. Moreover, compared with the raw high-resolution traffic videos, the density map formed by the extracted features has a smaller size and thus experiences a shorter transmission time. For the edge server, we propose two vehicle counting models, termed F2F-M and O2O-M, by incorporating the temporal correlation of videos captured from multiple city cameras for performance enhancement. Then, we integrate the lightweight feature extraction scheme and vehicle counting models to obtain two end-to-end vehicle counting models, including Lite-F2F-M and Lite-O2O-M. Finally, through comprehensive real-data experiments, we demonstrate that our Lite-F2F-M and Lite-O2O-M models are superior to the state-of-the-art in terms of vehicle counting performance and time efficiency. Our multifold contributions are addressed as follows.

- To the best of our knowledge, this is the first work to design a CityCam-to-Edge cooperative learning framework by leveraging collaboration between the city cameras and the edge server for vehicle counting.
- To simultaneously reduce the city cameras' memory consumption and the data transmission time, we design a lightweight feature extraction scheme by employing multiple convolutional layers, each of which has few kernels in deep learning architecture.
- To improve vehicle counting performance, we propose F2F-M and O2O-M models, taking into account the temporal correlation among videos from multiple city cameras.
- To validate the advantages of our schemes, we set up a series of real-data experiments to evaluate vehicle counting performance and time efficiency via the comparison with the state-of-the-art.

The remainder of this paper is organized as follows. Related works are briefly summarized in Section II. We detail our proposed models in Section III. In Section IV, we conduct real-data experiments and analyze all results. Finally, Section V concludes this paper and discusses our future work.

II. Related Works

The most recent methods of magnetic sensor-based vehicle counting and vision-based vehicle counting are summarized in the following.

A. Magnetic Sensor-Based Vehicle Counting Methods

In vehicle counting methods, 2-axis magnetic sensors and 3-axis magnetic sensors are mainly used. The works of [26]–[28] analyzed the fluctuations captured from 2-axis magnetic sensors for vehicle counting. An enhanced vehicle counting method was proposed by [31] to process 3-axis magnetic sensors' signal by considering a normalized cross-correlation of the 3-axis signal. Besides, the magnetic sensor-based vehicle counting methods have been applied for counting vehicles in parking lots by Andrius *et al.* [29] and in a street parking system by Zhang *et al.* [30]. However, the unreliability of magnetic sensor signal may cause a performance loss in these magnetic sensor-based vehicle counting approaches, and the deployment and maintenance of magnetic sensors usually bring lane closure, traffic disruption, and other inconvenience to people's daily life.

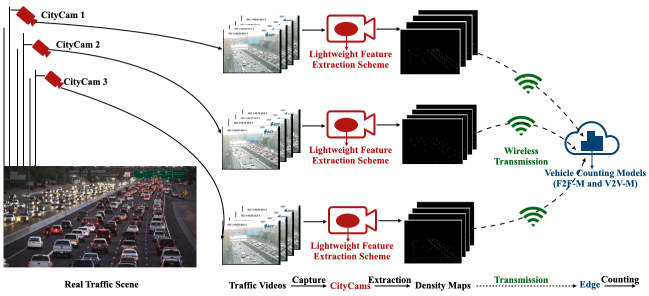


Fig. 1. Our CityCam-to-Edge cooperative learning framework.

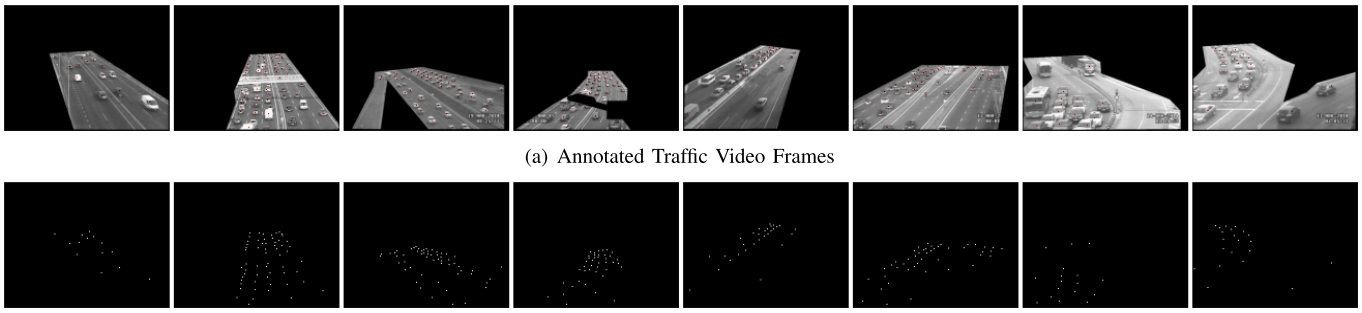
B. Vision-Based Vehicle Counting Methods

Compared with magnetic sensor-based vehicle counting methods, vision-based methods that utilize city cameras' data yield less traffic disruption during installation and maintenance. Also, the city cameras' data can provide more details for better understanding traffic flow [44]–[46]. Frame differencing methods count vehicles by analyzing the difference between sequential frames [32], [33], which suffer the changes of abrupt illumination and background. Detection-based methods count vehicles by detecting vehicles in each frame [34], [35] and thus have poor performance when being applied to high-occlusion videos. Tracking-based methods count vehicles by tracking each vehicle in each frame [36]–[39], which fail with low-frame rate videos due to a lack of motion information. Density estimation-based methods are proposed to count vehicles by estimating pixel-level density of images [40], [47], but they suffer from low accuracy when the images are large-perspective. However, these aforementioned conventional vehicle counting models cannot work well when video frames are high-occlusion, high-resolution and large-perspective. Recently, driven by the explosive progress of learning methodology, promising deep learning-based methods are used to process high-occlusion, high-resolution and large-perspective video frames. Nevertheless, since the deep learning-based models should be implemented on a server, expensive costs of data transmission, data computation, and device memory may be consumed.

To reduce the costs of processing traffic videos, a CityCam-to-Edge cooperative learning framework is designed as a novel solution to the problem of vehicle counting by exploiting lightweight feature extraction and inter-frame temporal correlation in traffic videos.

III. CityCam-to-Edge Cooperative Learning

Our proposed CityCam-to-Edge cooperative learning framework is presented in Fig. 1. In this framework, the city cameras



(a) Annotated Traffic Video Frames

(b) Extracted Density Maps based on FCN

Fig. 2. Examples of annotated traffic video frames and their corresponding density maps.

capture traffic videos to record the real traffic scene and then run the lightweight feature extraction scheme to extract density maps from the traffic videos for transmission, and the edge server implements the F2F-M and O2O-M vehicle counting models by processing the extracted density maps received from the city cameras. The challenges in the design of our proposed frameworks lie in two aspects: i) an effective and lightweight neural network architecture should be developed for feature extraction while reducing the utilization of parameters; and ii) the cooperation of multiple city cameras should be leveraged to enhance the accuracy of vehicle counting. The details of our design are addressed in the following.

A. Lightweight Feature Extraction Scheme

The existing object counting methods [43], [48]–[50] firstly estimate the object density map of one image and then directly sum the density of every pixel in the whole image to obtain the object counts. Motivated by these works, our feature extraction scheme is designed to extract the density maps of images for vehicle counting. Fully convolutional network (FCN) [51] is a popular deep neural network to map images into density maps and is used in our feature extraction scheme. Several examples of annotated traffic video frames and their corresponding extracted density maps based on FCN are shown in Fig. 2.

Assume that there are $\mathbf{N}$ blocks in FCN and two convolutional layers in each block. Let c_i^j be the number of input channels of the j-th convolutional layer in the i-th block, s_i^j be the kernel size of the j-th convolutional layer, k_i^j be the number of kernels of the j-th convolutional layer, where $j \in \{1, 2\}$. The number of kernels in a layer is equal to the number of input channels in its next layer, *i.e.,* $k_i^1 = c_i^2$. Accordingly, we can calculate the total number of parameters, P_{FCN}, in FCN as below:

$$P_{FCN} = \sum_{i=1}^{\mathbf{N}} c_i^1 s_i^1 k_i^1 + c_i^2 s_i^2 k_i^2 = \sum_{i=1}^{\mathbf{N}} c_i^1 s_i^1 k_i^1 + k_i^1 s_i^2 k_i^2. \quad (1)$$

A large number of parameters in FCN consumes too much memory resource of city cameras. Considering the memory limitation of city cameras, in this paper, we want to design a lightweight FCN model with relatively fewer parameters for extracting density maps in order to lower the city cameras' memory consumption. In [52], Lan *et al.* proposed a lightweight word embedding model by adding one convolutional layer with few kernels in the original word embedding learning architecture to reduce the utilization of parameters while maintaining the word embedding performance. By extending this idea to a more general scenario, our scheme applies multiple convolutional layers with few kernels in the architecture of FCN to obtain lightweight FCN for density map extraction.

Technically speaking, according to the theoretical guidance that the number of parameters in a lightweight FCN model should not be larger than the number of parameters in the original FCN model, we propose 4 rules to devise our lightweight FCN in this paper. i) Multiple convolutional layers with few kernels are added between two convolutional layers from the 1st block to the third last block of the original FCN. ii) The additional convolutional layers in the same block have the same number of kernels. iii) The number of kernels of additional convolutional layers in each block is respectively set following the same ratio. For example, if the number of kernels in the first layer of the 2nd block is twice as much as the number of kernels in the first layer of the 1st block, the number of kernels in the additional layer of the 2nd block is set twice as much as the number of kernels in the additional layer of 1st block. iv) The kernels in all additional convolutional layers have the same size.

In the original FCN architecture, two deconvolutional layers are used in the last two blocks (*i.e.*, the $(N-1)$-th block and the N-th block). Thus, to maintain the functions of convolutional layers and deconvolutional layers, we only add additional convolutional layers from the 1st block to the $(N-2)$-th block in our proposed lightweight FCN. According to the aforementioned rules, we add $\mathbf{L}$ convolutional layers with $\frac{k_i^1}{k_1^1}\mathbf{K}$ kernels in the i-th block ($i \in [1, \mathbf{N-2}]$). Denote s_a as the size of kernels in additional convolutional layers. Since the number of kernels in a layer is equal to the number of input channels in its next layer, we can easily compute the following information in our lightweight FCN: i) the number of parameters of the first convolutional layer in the i-th block is $c_i^1 s_i^1 k_i^1$; ii) the number of parameters of the first additional convolutional layer in the i-th block is $k_i^1 s_a \frac{k_i^1}{k_1^1}\mathbf{K}$; iii) the total number of parameters of the rest of $\mathbf{L}-1$ additional convolutional layers in the i-th block is $\frac{k_i^1}{k_1^1}\mathbf{K} s_a \frac{k_i^1}{k_1^1}\mathbf{K}(\mathbf{L}-1)$; iv) the parameters of the last convolutional layer in the i-th block are $\frac{k_i^1}{k_1^1}\mathbf{K} s_i^2 k_i^2$ since $c_i^2 = \frac{k_i^1}{k_1^1}\mathbf{K}$. Accordingly, we can get the

TABLE I
THE ARCHITECTURE OF FULLY CONVOLUTIONAL NETWORK (FCN)

Block	Layers	
1	$3 \times 3 \times 64$ Conv1_1 (ReLU)	$3 \times 3 \times 64$ Conv1_2 (ReLU)
2	$3 \times 3 \times 128$ Conv2_1 (ReLU)	$3 \times 3 \times 128$ Conv2_2 (ReLU)
3	$3 \times 3 \times 256$ Conv3_1 (ReLU)	$3 \times 3 \times 256$ Conv3_2 (ReLU)
4	$3 \times 3 \times 256$ Conv4_1 (ReLU)	$3 \times 3 \times 256$ Conv4_2 (ReLU)
5	$3 \times 3 \times 256$ Conv5_1 (ReLU)	$3 \times 3 \times 512$ Conv5_2 (ReLU)
6	$3 \times 3 \times 512$ Conv6_1 (ReLU)	$3 \times 3 \times 512$ Conv6_2 (ReLU)
7	$3 \times 3 \times 512$ Conv7_1 (ReLU)	$3 \times 3 \times 256$ Deconv7_2 (ReLU)
8	$3 \times 3 \times 64$ Deconv8_1 (ReLU)	$1 \times 1 \times 1$ Conv8_2 (ReLU)

TABLE II
EXAMPLES OF OUR LIGHTWEIGHT FCN

Block	Layers		
1	$3 \times 3 \times 64$ Conv1_1 (ReLU)	$\mathbf{3 \times 3 \times K}$ **Conv1_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 64$ Conv1_3 (ReLU)
2	$3 \times 3 \times 128$ Conv2_1 (ReLU)	$\mathbf{3 \times 3 \times 2K}$ **Conv2_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 128$ Conv2_3 (ReLU)
3	$3 \times 3 \times 256$ Conv3_1 (ReLU)	$\mathbf{3 \times 3 \times 4K}$ **Conv3_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 256$ Conv3_3 (ReLU)
4	$3 \times 3 \times 256$ Conv4_1 (ReLU)	$\mathbf{3 \times 3 \times 4K}$ **Conv4_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 256$ Conv4_3 (ReLU)
5	$3 \times 3 \times 256$ Conv5_1 (ReLU)	$\mathbf{3 \times 3 \times 4K}$ **Conv5_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 512$ Conv5_3 (ReLU)
6	$3 \times 3 \times 512$ Conv6_1 (ReLU)	$\mathbf{3 \times 3 \times 8K}$ **Conv6_2 (ReLU)** $\times \mathbf{L}$	$3 \times 3 \times 512$ Conv6_3 (ReLU)
7	$3 \times 3 \times 512$ Conv7_1 (ReLU)	$3 \times 3 \times 256$ Deconv7_2 (ReLU)	
8	$3 \times 3 \times 64$ Deconv8_1 (ReLU)	$1 \times 1 \times 1$ Conv8_2 (ReLU)	

total number of parameters, P_{Lite}, in our lightweight FCN as follows:

$$P_{Lite} = \sum_{i=1}^{\mathbf{N-2}} c_i^1 s_i^1 k_i^1 + k_i^1 s_a \frac{k_i^1}{k_1^1}\mathbf{K} + \frac{k_i^1}{k_1^1}\mathbf{K} s_a \frac{k_i^1}{k_1^1}\mathbf{K}(\mathbf{L}-1) + \frac{k_i^1}{k_1^1}\mathbf{K} s_i^2 k_i^2 + \sum_{i=\mathbf{N}-1}^{\mathbf{N}} c_i^1 s_i^1 k_i^1 + k_i^1 s_i^2 k_i^2. \quad (2)$$

In order to ensure that the number of parameters in the lightweight FCN is not larger than that in FCN, we have $P_{Lite} \leq P_{FCN}$, which can be equivalently rewritten in Eq. (3) according to Eq. (1) and Eq. (2).

$$\sum_{i=1}^{\mathbf{N-2}} k_i^1 s_a \frac{k_i^1}{k_1^1}\mathbf{K} + \frac{k_i^1}{k_1^1}\mathbf{K} s_a \frac{k_i^1}{k_1^1}\mathbf{K}(\mathbf{L}-1) + \frac{k_i^1}{k_1^1}\mathbf{K} s_i^2 k_i^2 - k_i^1 s_i^2 k_i^2 \leq 0. \quad (3)$$

Theoretically, we can design our lightweight FCN based on any one kind of FCN using the proposed 4 rules. In this paper, we take the architecture of FCN in Table I as an example for density map extraction. Correspondingly, the architecture of our lightweight FCN is presented in Table II, where we use the 3×3 kernel for all additional convolutional layers (i.e. $s_a = 9$). From Table I, we know that $\mathbf{N} = 8$, $s_1^2 = s_2^2 = s_3^2 = s_4^2 = s_5^2 = s_6^2 = 9$, $k_1^1 = 64$, $k_2^1 = 128$, $k_3^1 = 256$, $k_4^1 = 256$, $k_5^1 = 256$, $k_6^1 = 512$, $k_1^2 = 64$, $k_2^2 = 128$, $k_3^2 = 256$, $k_4^2 = 256$, $k_5^2 = 512$, and $k_6^2 = 512$. Then, according to Eq. (3), Eq. (4) should be satisfied to guarantee that the number of parameters of our lightweight FCN is not higher than that of the original FCN, with which especially, the values of $\mathbf{K}$ and $\mathbf{L}$ can be set properly and flexibly.

$$1053(\mathbf{L}-1)\mathbf{K}^2 + 144000\mathbf{K} - 4902912 \leq 0. \quad (4)$$

By solving Eq. (4), we further have:

$$\mathbf{L} \leq \lceil \frac{4902912 - 144000\mathbf{K}}{1053\mathbf{K}^2} \rceil + 1. \quad (5)$$

Eq. (5) is used as a theoretical constraint to appropriately set the values of $\mathbf{K}$ and $\mathbf{L}$ to design the architecture of our lightweight FCN models, for which experiment settings and results are demonstrated in Section IV-A.3.

B. Vehicle Counting Models

Through exploiting the temporal correlation among sequential traffic video frames, multiple city cameras can help each other to accomplish vehicle counting with an enhanced performance. To formulate such a temporal correlation, Long Short-Term Memory (LSTM) [53], which can maintain internal hidden states to model the dynamic temporal behavior of sequences, is adopted in our models. Let $LSTM(X; \Phi)$ represent LSTM function, where X denotes input and Φ denotes all parameters used in LSTM neural network. Formally, LSTM is implemented by the following recurrent functions:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right), \quad (6)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \quad (7)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \quad (8)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right), \quad (9)$$

$$h_t = o_t \tanh\left(c_t\right), \quad (10)$$

where $\sigma(\cdot)$ is the logistic sigmoid function, $\tanh(\cdot)$ is the tanh activation function, x is input, and i, f, c, and o are respectively the input gate, forget gate, memory cell and output gate activation vectors, the size of which is the same as hidden vector h. And W_{xi}, W_{hi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xc}, W_{hc}, W_{xo}, W_{ho}, and W_{co} are weight matrices, and b_i, b_f, b_c, b_o are bias terms.

With the LSTM function, we can obtain the final hidden vector and map it into a 1-dimension count label by using a fully connected layer to realize a supervised vehicle counting learning. This whole process is denoted as $FC(LSTM(X; \Phi); \Theta)$, where $FC(\cdot)$ is the function of fully connected layer, and Θ represents the parameters of $FC(\cdot)$. Next, $FC(LSTM(X; \Phi); \Theta)$ is utilized in our F2F-M and O2O-M models for vehicle counting.

1) F2F-M Model: F2F-M model utilizes the temporal correlation among videos captured from multiple city cameras in a frame-to-frame manner. Suppose there are n consecutive cameras arranged along a road. Let $\{t_1, t_2, \ldots, t_n\}$ be a sequential time series and $I_{t_i}^i$ ($i \in [1, n]$) be the frame captured by the i-th city camera at time t_i. These n frames have similar vehicle scenes, which can be used to depict the temporal correlation among videos and processed by $FC(LSTM(X; \Phi); \Theta)$ for vehicle counting.

First of all, the frame $I_{t_i}^i$ is put into $FC(LSTM(X; \Phi); \Theta)$ to obtain the predicted vehicle count $C_{t_i}^i$, *i.e.*,

$$C_{t_i}^i = FC(LSTM(I_{t_i}^i; \Phi); \Theta). \quad (11)$$

Since we have the groundtruth vehicle count, ${}_0C_{t_i}^i$, for the frame $I_{t_i}^i$ in the dataset, we can define the loss function of vehicle counting as:

$$L_{C_{t_i}^i} = ||C_{t_i}^i - {}_0C_{t_i}^i||_2^2. \quad (12)$$

As there are n frames as input, finally, we can obtain the overall loss function of F2F-M model:

$$L_{F2F-M} = \sum_{i=1}^{n} L_{C_{t_i}^i}, \tag{13}$$

which will be minimized via Adam [54] for training F2F-M model.

2) O2O-M Model: O2O-M model exploits the temporal correlation among videos captured from multiple city cameras in a video-to-video manner. Similarly, suppose there are n consecutive cameras arranged along a road, and one traffic video consists of m frames that exist temporal correlation. Denote $I_{t_{i+j-1}}^i$ as the j-th frame in the video captured by the i-th camera at time t_{i+j-1} ($i \in [1, n], j \in [1, m]$). Then, we can represent $\{I_{t_1}^1, I_{t_2}^1, \ldots, I_{t_{n+m-1}}^n\}$ as n videos that show temporal correlated vehicle scenes captured by these n consecutive cameras. In O2O-M model, $\{I_{t_1}^1, I_{t_2}^1, \ldots, I_{t_{n+m-1}}^n\}$ is processed by $FC(LSTM(X; \Phi); \Theta)$ for vehicle counting. The loss function of O2O-M model is similar to that of F2F-M model. While, the main different places are that the input contains n videos and each video consists of m frames. Thus, the overall loss function of O2O-M model is formulated as:

$$L_{O2O-M} = \sum_{i=1}^{n} \sum_{j=1}^{m} L_{C_{t_{i+j-1}}^i}, \tag{14}$$

which is also minimized via Adam to train O2O-M model.

C. End-to-End Vehicle Counting Models

In this section, we propose two end-to-end vehicle counting models, Lite-F2F-M and Lite-O2O-M, by combining our proposed lightweight FCN mentioned in Section III-A and two vehicle counting models, F2F-M and O2O-M, introduced in Section III-B.

1) Lite-F2F-M Model: Lite-F2F-M model is based on F2F-M model. Compared with F2F-M, the different point is that the lightweight feature extraction scheme is used as a data preprocessing method for vehicle counting in Lite-F2F-M. Denote our lightweight FCN as $Lite(X; \Gamma)$, where X represents input and Γ represents the parameters of our lightweight FCN. In Lite-F2F-M, the input also has n frames represented by $\{I_{t_1}^1, I_{t_2}^2, \ldots, I_{t_n}^n\}$. Then, the density map $F_{t_i}^i$ should firstly be extracted from the frame $I_{t_i}^i$ through our lightweight FCN, *i.e.*,

$$F_{t_i}^i = Lite(I_{t_i}^i; \Gamma). \tag{15}$$

Since we have the groundtruth density map in the dataset, the loss function of density map estimation for $I_{t_i}^i$ is defined as follows:

$$L_{D_{t_i}^i} = ||F_{t_i}^i - {}_0F_{t_i}^i||_2^2, \tag{16}$$

where ${}_0F_{t_i}^i$ is the groundtruth density map of $I_{t_i}^i$. Then, for vehicle counting, the density map $F_{t_i}^i$ should be put into $FC(LSTM(X; \Phi); \Theta)$ to get the predicted vehicle count $C_{t_i}^i$ for $I_{t_i}^i$; that is,

$$C_{t_i}^i = FC(LSTM(F_{t_i}^i; \Phi); \Theta). \tag{17}$$

Given the groundtruth vehicle count annotated in the dataset, the loss function of vehicle counting is defined to be:

$$L_{C_{t_i}^i} = ||C_{t_i}^i - {}_0C_{t_i}^i||_2^2. \tag{18}$$

With n frames as input, the overall loss function will be:

$$L_{Lite-F2F-M} = \sum_{i=1}^{n} L_{D_{t_i}^i} + \lambda \sum_{i=1}^{n} L_{C_{t_i}^i}, \tag{19}$$

where λ is the weight to balance two loss terms. The overall loss function of Lite-F2F-M will also be minimized via Adam. Algorithm III-C.1 outlines the pseudo code of the training process of Lite-F2F-M model.

Algorithm 1 Lite-F2F-M Training Algorithm

Input: n frames: $\{I_{t_1}^1, I_{t_2}^2, \ldots, I_{t_n}^n\}$
Output: parameters of $Lite(\cdot)$, $LSTM(\cdot)$, $FC(\cdot)$: Γ, Φ, Θ.
for $i = 1$ to n **do**
  $F_{t_i}^i = Lite(I_{t_i}^i; \Gamma)$
  $L_{D_{t_i}^i} = ||F_{t_i}^i - {}_0F_{t_i}^i||_2^2$
  $C_{t_i}^i = FC(LSTM(F_{t_i}^i; \Phi); \Theta)$
  $L_{C_{t_i}^i} = ||C_{t_i}^i - {}_0C_{t_i}^i||_2^2$
end for
$L_{Lite-F2F-M} = \sum_{i=1}^{n} L_{D_{t_i}^i} + \lambda \sum_{i=1}^{n} L_{C_{t_i}^i}$
$\Gamma, \Phi, \Theta \leftarrow Adam(L, \Gamma, \Phi, \Theta)$

2) Lite-O2O-M Model: On the basis of O2O-M model, we build Lite-O2O-M model, where n videos are first put into our lightweight FCN to obtain the density maps and then processed for vehicle counting. The input of Lite-O2O-M are n videos denoted by $\{I_{t_1}^1, I_{t_2}^1, \ldots, I_{t_{n+m-1}}^n\}$. Similar to the overall formulation of Lite-F2F-M model, the overall loss function of Lite-O2O-M model is expressed by Eq. (20), which is also minimized via Adam.

$$L_{Lite-O2O-M} = \sum_{i=1}^{n} \sum_{j=1}^{m} L_{D_{t_{i+j-1}}^i} + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} L_{C_{t_{i+j-1}}^i}. \tag{20}$$

Algorithm III-C.1 presents the pseudo code of the training process of Lite-O2O-M.

Algorithm 2 Lite-O2O-M Training Algorithm

Input: n videos: $\{I_{t_1}^1, I_{t_2}^1, \ldots, I_{t_{n+m-1}}^n\}$
Output: parameters of $Lite(\cdot)$, $LSTM(\cdot)$, $FC(\cdot)$: Γ, Φ, Θ.
for $i = 1$ to n **do**
  for $j = 1$ to m **do**
    $F_{t_{i+j-1}}^i = Lite(I_{t_{i+j-1}}^i; \Gamma)$
    $L_{D_{t_{i+j-1}}^i} = ||F_{t_{i+j-1}}^i - {}_0F_{t_{i+j-1}}^i||_2^2$
    $C_{t_{i+j-1}}^i = FC(LSTM(F_{t_{i+j-1}}^i; \Phi); \Theta)$
    $L_{C_{t_{i+j-1}}^i} = ||C_{t_{i+j-1}}^i - {}_0C_{t_{i+j-1}}^i||_2^2$
  end for
end for
$L_{Lite-O2O-M} = \sum_{i=1}^{n} \sum_{j=1}^{m} L_{D_{t_{i+j-1}}^i} + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} L_{C_{t_{i+j-1}}^i}$
$\Gamma, \Phi, \Theta \leftarrow Adam(L, \Gamma, \Phi, \Theta)$

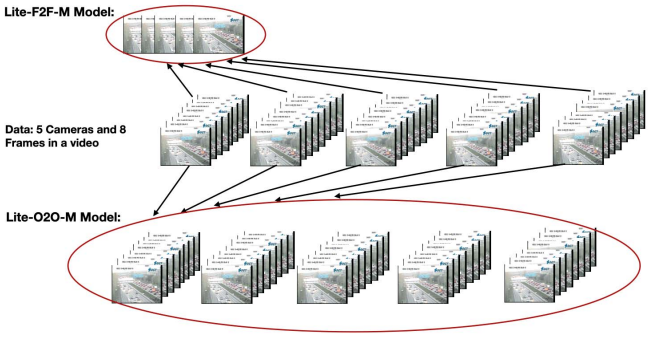


Fig. 3. An example for illustrating differences between Lite-F2F-M model and Lite-O2O-M model.

D. Model Comparison

An illustrative example is presented in Fig. 3 to compare the difference between Lite-F2F-M model and Lite-O2O-M model from the perspective of data process.

Assume that five consecutive cameras are arranged along a road and every video captured by each camera contains 8 frames. In our Lite-F2F-M model, we use one frame from each video (*i.e.*, totally, 5 frames) in every training round, in which the temporal information among these frames is leveraged to improve the accuracy of vehicle counting. Differently, in our Lite-O2O-M model, we employ 5 videos in every training round. Accordingly, not only the temporal information among the 5 videos but also the temporal information among the 8 frames in one video can be exploited to enhance the accuracy of vehicle counting.

IV. Experiments

In this section, extensive experiments are conducted to validate the superiority of our proposed Lite-F2F-M and Lite-O2O-M models over the state-of-the-art in vehicle counting performance and time efficiency. The codes of our model implementation are now available at https://github.com/ahahnut/Lite-F2F-V2V-M.

A. Experiment Settings

The dataset, baselines, our models' architectures are described below.

1) Dataset: TRANCOS [43] contains 1244 images of traffic scenes collected from surveillance camera videos with 46796 annotated vehicles in total. Since we take the cooperation of multiple city cameras into account, we create our own dataset from TRANCOS to satisfy the need of our application scenario. In the experiments, we consider 5 city cameras that record the traffic information along a road. For the images that show a similar vehicle scene, we treat every 8 image frames as one video, pick 40 images to form 5 videos, and assign one video to each camera's records. Totally, there are 23 videos in each city camera. This default setting has been used as an example to illustrate the difference between our Lite-F2F-M model and our Lite-O2O-M model in Section III-D.

Moreover, other different settings are configured in the following experiments to study the influence of camera density and inter-frame time difference on our two proposed models in Section IV-D and Section IV-E, respectively.

TABLE III
SETTINGS AND PARAMETERS OF LIGHTWEIGHT FCN IN EXPERIMENTS

Number of Parameters: 12.49×10^6 (FCN)			
Number of Parameters (Lightweight FCN)			
Kernels \ Layers	$\mathbf{L}=1$	$\mathbf{L}=2$	$\mathbf{L}=3$
$\mathbf{K}=8$	8.74×10^6 (Lite1_1)	8.81×10^6 (Lite1_2)	8.88×10^6 (Lite1_3)
$\mathbf{K}=12$	9.32×10^6 (Lite2_1)	9.47×10^6 (Lite2_2)	9.62×10^6 (Lite2_3)
$\mathbf{K}=16$	9.89×10^6 (Lite3_1)	10.16×10^6 (Lite3_2)	10.43×10^6 (Lite3_3)

TABLE IV
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE1_1)

Block	Layers		
1	$3 \times 3 \times 64$ Conv1_1 (ReLU)	$\mathbf{3 \times 3 \times 8}$ **Conv1_2 (ReLU)**	$3 \times 3 \times 64$ Conv1_3 (ReLU)
2	$3 \times 3 \times 128$ Conv2_1 (ReLU)	$\mathbf{3 \times 3 \times 16}$ **Conv2_2 (ReLU)**	$3 \times 3 \times 128$ Conv2_3 (ReLU)
3	$3 \times 3 \times 256$ Conv3_1 (ReLU)	$\mathbf{3 \times 3 \times 32}$ **Conv3_2 (ReLU)**	$3 \times 3 \times 256$ Conv3_3 (ReLU)
4	$3 \times 3 \times 256$ Conv4_1 (ReLU)	$\mathbf{3 \times 3 \times 32}$ **Conv4_2 (ReLU)**	$3 \times 3 \times 256$ Conv4_3 (ReLU)
5	$3 \times 3 \times 256$ Conv5_1 (ReLU)	$\mathbf{3 \times 3 \times 32}$ **Conv5_2 (ReLU)**	$3 \times 3 \times 512$ Conv5_3 (ReLU)
6	$3 \times 3 \times 512$ Conv6_1 (ReLU)	$\mathbf{3 \times 3 \times 64}$ **Conv6_2 (ReLU)**	$3 \times 3 \times 512$ Conv6_3 (ReLU)
7	$3 \times 3 \times 512$ Conv7_1 (ReLU)	$3 \times 3 \times 256$ Deconv7_2 (ReLU)	
8	$3 \times 3 \times 64$ Deconv8_1 (ReLU)	$1 \times 1 \times 1$ Conv8_2 (ReLU)	

2) Baselines: We compare our proposed models with two baseline schemes.

- Baseline 1. In [47], the authors developed a deep learning model to do vehicle counting based on FCN. They designed a supervised learning model to learn the vehicle counts from density maps extracted by FCN without considering the temporal information among video frames.
- Baseline 2. In [41], the authors proposed a deep learning model based on FCN-LSTM. They improved the baseline 1 model by considering the temporal information among video frames with the help of LSTM, which is the state-of-the-art approach for vehicle counting.

3) Architecture of Lightweight FCN: According to the idea of our proposed lightweight feature extraction scheme in Section III-A, we set $\mathbf{K} = \{8, 12, 16\}$ and $\mathbf{L} = \{1, 2, 3\}$, the settings of which satisfy Eq. (5) to get 9 architectures of lightweight FCN for experiments. In Table III, we list these 9 settings of lightweight FCN, termed Lite1_1, Lite1_2, Lite1_3, Lite2_1, Lite2_2, Lite2_3, Lite3_1, Lite3_2, Lite3_3, and calculate the number of parameters used in these nine lightweight architectures. From Table III, we can see that our proposed 9 lightweight architectures indeed decrease the utilization of parameters compared with the original FCN. Besides, we present the details of these 9 lightweight architectures in Table IV-Table XII.

Actually, according to Eq. (5) in our lightweight feature extraction scheme, we can obtain a number of lightweight architectures by properly choosing the values of $\mathbf{K}$ and $\mathbf{L}$. Due to the page limitation, only nine of them are presented for performance evaluation in this paper.

4) Architecture of F2F-M and O2O-M: Both F2F-M model and O2O-M model have the same deep learning architecture. Specifically, we use 3 LSTM layers and set the size of hidden vector h as 100. In addition, one FC layer is used after 3 LSTM layers to map the final 100-dimension hidden vector into a 1-dimension count label.

TABLE V
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE1_2)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 8 Conv1_2 (ReLU) ×2**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 16 Conv2_2 (ReLU) ×2**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 32 Conv3_2 (ReLU) ×2**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 32 Conv4_2 (ReLU) ×2**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 32 Conv5_2 (ReLU) ×2**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 64 Conv6_2 (ReLU) ×2**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE VI
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE1_3)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 8 Conv1_2 (ReLU) ×3**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 16 Conv2_2 (ReLU) ×3**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 32 Conv3_2 (ReLU) ×3**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 32 Conv4_2 (ReLU) ×3**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 32 Conv5_2 (ReLU) ×3**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 64 Conv6_2 (ReLU) ×3**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE VII
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE2_1)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 12 Conv1_2 (ReLU)**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 24 Conv2_2 (ReLU)**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 48 Conv3_2 (ReLU)**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 48 Conv4_2 (ReLU)**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 48 Conv5_2 (ReLU)**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 96 Conv6_2 (ReLU)**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE VIII
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE2_2)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 12 Conv1_2 (ReLU) ×2**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 24 Conv2_2 (ReLU) ×2**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 48 Conv3_2 (ReLU) ×2**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 48 Conv4_2 (ReLU) ×2**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 48 Conv5_2 (ReLU) ×2**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 96 Conv6_2 (ReLU) ×2**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE IX
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE2_3)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 12 Conv1_2 (ReLU) ×3**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 24 Conv2_2 (ReLU) ×3**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 48 Conv3_2 (ReLU) ×3**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 48 Conv4_2 (ReLU) ×3**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 48 Conv5_2 (ReLU) ×3**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 96 Conv6_2 (ReLU) ×3**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE X
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE3_1)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 16 Conv1_2 (ReLU)**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 32 Conv2_2 (ReLU)**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 64 Conv3_2 (ReLU)**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 64 Conv4_2 (ReLU)**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 64 Conv5_2 (ReLU)**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 128 Conv6_2 (ReLU)**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

5) Architectures of Lite-F2F-M and Lite-O2O-M: By combining F2F-M or O2O-M with 9 lightweight FCN architectures Litex_y where $x, y \in \{1, 2, 3\}$, we can obtain 18 end-to-end vehicle counting methods as demonstrated in Fig. 4 for comprehensive performance evaluation.

TABLE XI
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE3_2)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 16 Conv1_2 (ReLU) ×2**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 32 Conv2_2 (ReLU) ×2**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 64 Conv3_2 (ReLU) ×2**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 64 Conv4_2 (ReLU) ×2**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 64 Conv5_2 (ReLU) ×2**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 128 Conv6_2 (ReLU) ×2**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

TABLE XII
THE ARCHITECTURE OF LIGHTWEIGHT FCN (LITE3_3)

Block	Layers		
1	3 × 3 × 64 Conv1_1 (ReLU)	**3 × 3 × 16 Conv1_2 (ReLU) ×3**	3 × 3 × 64 Conv1_3 (ReLU)
2	3 × 3 × 128 Conv2_1 (ReLU)	**3 × 3 × 32 Conv2_2 (ReLU) ×3**	3 × 3 × 128 Conv2_3 (ReLU)
3	3 × 3 × 256 Conv3_1 (ReLU)	**3 × 3 × 64 Conv3_2 (ReLU) ×3**	3 × 3 × 256 Conv3_3 (ReLU)
4	3 × 3 × 256 Conv4_1 (ReLU)	**3 × 3 × 64 Conv4_2 (ReLU) ×3**	3 × 3 × 256 Conv4_3 (ReLU)
5	3 × 3 × 256 Conv5_1 (ReLU)	**3 × 3 × 64 Conv5_2 (ReLU) ×3**	3 × 3 × 512 Conv5_3 (ReLU)
6	3 × 3 × 512 Conv6_1 (ReLU)	**3 × 3 × 128 Conv6_2 (ReLU) ×3**	3 × 3 × 512 Conv6_3 (ReLU)
7	3 × 3 × 512 Conv7_1 (ReLU)	3 × 3 × 256 Deconv7_2 (ReLU)	
8	3 × 3 × 64 Deconv8_1 (ReLU)	1 × 1 × 1 Conv8_2 (ReLU)	

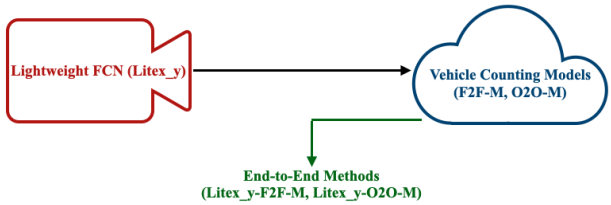


Fig. 4. End-to-End vehicle counting methods in experiments.

B. Evaluation of Vehicle Counting Performance

We train the two baseline models and our Lite-F2F-M and Lite-O2O-M models by using the default dataset setting described in Section IV-A.1 with the learning rate $lr = 0.001$, the balance weight of two loss terms $\lambda = 0.001$, and the number of training epochs $ep = 150$. After that, we use well-trained models to count the vehicles in 40 frames for testing. The mean absolute error (MAE) is calculated based on the groundtruth counts and the predicted counts. A smaller MAE value indicates a better counting performance. All the results are reported in Table XIII, which shows that there are 4 end-to-end methods (including Lite2_1-F2F-M, Lite2_2-F2F-M, Lite3_3-F2F-M, and Lite3_1-O2O-M) more accurate than the baseline models. Especially, Lite3_1-O2O-M has the best performance, which means that Lite-O2O-M model is more efficient than Lite-F2F-M model in terms of vehicle counting performance thanks to the consideration of the temporal correlation among videos captured from multiple city cameras in a video-to-video manner.

Furthermore, we investigate our proposed end-to-end vehicle counting methods from the aspects of counting accuracy, memory consumption, and cost-efficiency. i) The models' counting accuracy (Acc.) is computed as $(1 - MAE/GT) \times 100\%$, where GT denotes the ground-truth vehicle counts. ii) Since our lightweight feature extraction scheme aims to reduce the memory consumption in city cameras, the ratio of

TABLE XIII

MAE RESULTS OF METHODS TRAINED BY 5 CAMERAS AND 8 FRAMES IN ONE VIDEO (OURS V.S. BASELINES)

MAE:12.249 (Baseline 1)			
MAE:9.323 (Baseline 2)			
MAE (Lite-F2F-M)			
Layers / Kernels	$\mathbf{L}=1$	$\mathbf{L}=2$	$\mathbf{L}=3$
$\mathbf{K}=8$	12.747 (Lite1_1-F2F-M)	10.119 (Lite1_2-F2F-M)	15.789 (Lite1_3-F2F-M)
$\mathbf{K}=12$	**8.613 (Lite2_1-F2F-M)**	**7.324 (Lite2_2-F2F-M)**	12.712 (Lite2_3-F2F-M)
$\mathbf{K}=16$	9.845 (Lite3_1-F2F-M)	9.381 (Lite3_2-F2F-M)	**6.756 (Lite3_3-F2F-M)**
MAE (Lite-O2O-M)			
Layers / Kernels	$\mathbf{L}=1$	$\mathbf{L}=2$	$\mathbf{L}=3$
$\mathbf{K}=8$	11.645 (Lite1_1-O2O-M)	11.87 (Lite1_2-O2O-M)	12.993 (Lite1_3-O2O-M)
$\mathbf{K}=12$	9.64 (Lite2_1-O2O-M)	13.035 (Lite2_2-O2O-M)	14.556 (Lite2_3-O2O-M)
$\mathbf{K}=16$	**6.169 (Lite3_1-O2O-M)**	12.16 (Lite3_2-O2O-M)	13.202 (Lite3_3-O2O-M)

TABLE XIV

RESULTS OF ACC., PCT AND ACC./PCT (OURS V.S. BASELINES)

Methods	Acc.	Parameters	Memory	PCT	Acc./PCT
Baseline 1	67.438%	12.48×10^6	49.92MB	100%	0.67
Baseline 2	75.216%	12.48×10^6	49.92MB	100%	0.75
Lite2_1-F2F-M	77.104%	9.32×10^6	37.28MB	**74.6%**	1.03
Lite2_2-F2F-M	80.531%	9.47×10^6	37.88MB	75.81%	**1.06**
Lite3_3-F2F-M	82.041%	10.43×10^6	41.72MB	83.52%	0.98
Lite3_1-O2O-M	**83.601%**	9.89×10^6	39.56MB	79.21%	1.05

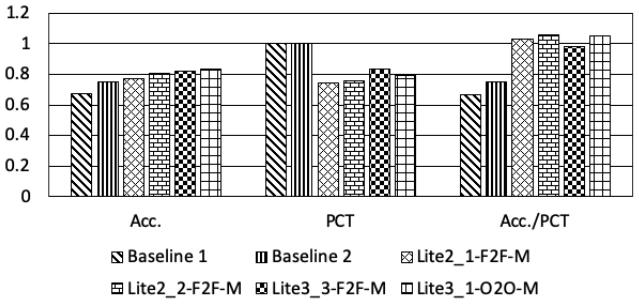


Fig. 5. Comparison results of Acc., PCT and Acc./PCT (Ours v.s. baselines).

the memory consumption in lightweight FCN to the memory consumption in the original FCN is used to measure the reduction of memory consumption. This ratio is denoted by PCT with a range of $[0, 100\%]$. A lower PCT value implies a larger decrease in memory consumption. iii) The ratio of counting accuracy to memory consumption is used to indicate the cost-efficiency for the compared schemes.

From the observations on Table XIV and Fig. 5, we can obtain three conclusions. i) Lite3_1-O2O-M method is the most efficient vehicle counting method since it has the highest counting accuracy. ii) Lite2_1-F2F-M method is the most cost-saving vehicle counting method due to its lowest PCT value. iii) Lite2_2-F2F-M method is the most cost-efficient vehicle counting method owing to its highest value of Acc./PCT.

C. Evaluation of Time Efficiency

In the two baseline models, the raw high-resolution traffic videos are directly sent from the city cameras to the edge server, which only costs data transmission time. While, in our proposed models, density maps are extracted from traffic videos and then sent from the city cameras to the edge server, which costs both the data extraction time and data transmission time. The raw data size, the density map size, and the data extraction time can be recorded when running vehicle counting models. IEEE 802.11p is adopted as a communication standard in vehicular networks, where the data is transmitted via wireless network in a short-range communication environment with the maximum wireless transmission speed at 27Mb/s [55]. The data transmission time equals the data size divided by transmission speed.

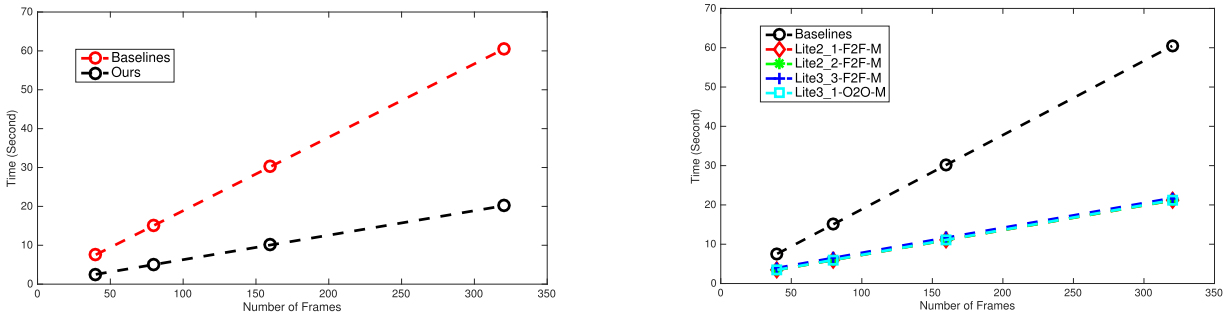
TABLE XV

TIME CONSUMPTION FOR DIFFERENT FRAMES (OURS V.S. BASELINES)

Methods	Frames	Data Transmission Time	Data Extraction Time	Total Time Consumption
Two Baselines	40	7.555s	/	7.555s
Two Baselines	80	15.111s	/	15.111s
Two Baselines	160	30.222s	/	30.222s
Two Baselines	320	60.444s	/	60.444s
Lite2_1-F2F-M	40	2.518s	0.914s	3.432s
Lite2_1-F2F-M	80	5.037s	0.942s	5.979s
Lite2_1-F2F-M	160	10.074s	0.98s	11.054s
Lite2_1-F2F-M	320	20.148s	0.987s	21.135s
Lite2_2-F2F-M	40	2.518s	0.937s	3.455s
Lite2_2-F2F-M	80	5.037s	0.964s	6.001s
Lite2_2-F2F-M	160	10.074s	0.994s	11.068s
Lite2_2-F2F-M	320	20.148s	1.001s	21.149s
Lite3_3-F2F-M	40	2.518s	1.452s	3.97s
Lite3_3-F2F-M	80	5.037s	1.503s	6.54s
Lite3_3-F2F-M	160	10.074s	1.604s	11.678s
Lite3_3-F2F-M	320	20.148s	1.606s	21.754s
Lite3_1-O2O-M	40	2.518s	0.96s	3.478s
Lite3_1-O2O-M	80	5.037s	0.987s	6.024s
Lite3_1-O2O-M	160	10.074s	1.046s	11.12s
Lite3_1-O2O-M	320	20.148s	1.085s	21.233s

In our experiments, when running our four efficient end-to-end vehicle counting methods analyzed in Section IV-B, including Lite2_1-F2F-M, Lite2_2-F2F-M, Lite3_3-F2F-M, and Lite3_1-O2O-M, on two Tesla V100-pcie GPUs, we record the time of extracting 40/80/160/320 frames and show the results in Table XV. After that, we calculate the data transmission time for 40/80/160/320 frames in baselines and our models. The total time consumption of the two baselines is equal to the data transmission time, while the total time consumption of our proposed models consists of data transmission time and data extraction time.

We present the data transmission time and total time consumption in Fig. 6 and Table XV, from which we obtain three critical findings. i) From Fig. 6a, we can see that the data transmission time of our models is much lower than that of two baselines, implying that our proposed Lite-F2F-M and Lite-O2O-M models greatly improve the data transmission efficiency. ii) In Fig. 6b, notice that the total time consumption of our models is also much lower than that of baselines even if there are the extra data extraction time in our models, which means the utilization of lightweight FCN in our models indeed greatly reduce the time consumption for vehicle counting by taking relatively little time to extract features. iii) The discrepancy of data transmission time/total time consumption between baselines and our models becomes larger with the increase of frames, indicating that our models will become more time efficient when transmitting a larger number of traffic videos.



(a) Data Transmission Time for 40/80/160/320 Frames (Ours v.s. Baselines) (b) Total Time Consumption for 40/80/160/320 Frames (Ours v.s. Baselines)

Fig. 6. Comparison result of time efficiency (Ours v.s. baselines).

TABLE XVI

MAE RESULTS OF OUR END-TO-END METHODS TRAINED BY 4 CAMERAS AND 8 FRAMES IN ONE VIDEO

MAE (Lite-F2F-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	15.847 (Lite1_1-F2F-M)	10.35 (Lite1_2-F2F-M)	16.706 (Lite1_3-F2F-M)
K = 12	12.22 (Lite2_1-F2F-M)	11.294 (Lite2_2-F2F-M)	15.348 (Lite2_3-F2F-M)
K = 16	10.273 (Lite3_1-F2F-M)	10.056 (Lite3_2-F2F-M)	**7.889 (Lite3_3-F2F-M)**
MAE (Lite-O2O-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	14.847 (Lite1_1-O2O-M)	12.75 (Lite1_2-O2O-M)	13.375 (Lite1_3-O2O-M)
K = 12	10.468 (Lite2_1-O2O-M)	13.869 (Lite2_2-O2O-M)	15.255 (Lite2_3-O2O-M)
K = 16	**6.569 (Lite3_1-O2O-M)**	12.516 (Lite3_2-O2O-M)	14.918 (Lite3_3-O2O-M)

TABLE XVII

MAE RESULTS OF OUR END-TO-END METHODS TRAINED BY 3 CAMERAS AND 8 FRAMES IN ONE VIDEO

MAE (Lite-F2F-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	16.001 (Lite1_1-F2F-M)	10.459 (Lite1_2-F2F-M)	16.862 (Lite1_3-F2F-M)
K = 12	12.739 (Lite2_1-F2F-M)	11.413 (Lite2_2-F2F-M)	16.266 (Lite2_3-F2F-M)
K = 16	10.916 (Lite3_1-F2F-M)	11.18 (Lite3_2-F2F-M)	**8.901 (Lite3_3-F2F-M)**
MAE (Lite-O2O-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	15.767 (Lite1_1-O2O-M)	13.427 (Lite1_2-O2O-M)	13.656 (Lite1_3-O2O-M)
K = 12	11.579 (Lite2_1-O2O-M)	14.22 (Lite2_2-O2O-M)	15.955 (Lite2_3-O2O-M)
K = 16	**7.553 (Lite3_1-O2O-M)**	12.801 (Lite3_2-O2O-M)	14.928 (Lite3_3-O2O-M)

D. Impact of City Camera Density

The MAE results of Lite-F2F-M and Lite-O2O-M models under the default dataset setting (*i.e.*, 5 city cameras arranged along the road and 8 frames in every video) are presented in Table XIII for analysis. In order to investigate the impact of city camera density on Lite-F2F-M and Lite-O2O-M models, we further train Lite-F2F-M and Lite-O2O-M models by taking into account two more different dataset settings: i) 4 city cameras arranged along the road and 8 frames in each video, and ii) 3 city cameras arranged along the road and 8 frames in each video. The corresponding MAE results are reported in Table XVI and Table XVII.

From Table XVI, we find that Lite3_3-F2F-M is the best architecture with the lowest MAE among 9 different architectures of Lite-F2F-M model and Lite3_1-O2O-M is the best architecture with the lowest MAE among 9 different architectures of Lite-O2O-M model. In Table XVII, Lite3_3-F2F-M is the best architecture of Lite-F2F-M model, and Lite3_1-O2O-M is the best architecture of Lite-O2O-M model.

Additionally, we draw the MAE results of Lite1_1-F2F-M, Lite1_3-F2F-M, Lite3_1-F2F-M, Lite1_1-O2O-M, Lite1_3-O2O-M, and Lite3_1-O2O-M in Fig. 7 for further analysis. Through observing Fig. 7, we can reach a conclusion that with the increase of city camera density, the temporal correlation among the captured videos becomes stronger, thereby enhancing the performance of Lite-F2F-M and Lite-O2O-M models.

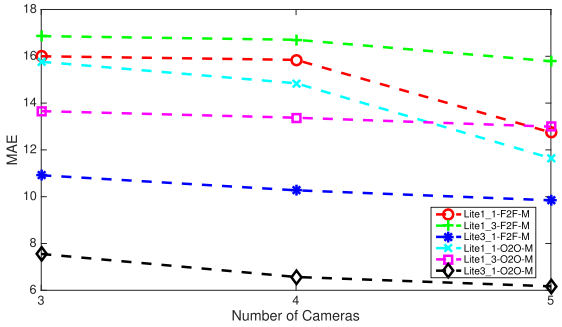


Fig. 7. Impact of city camera density on MAE.

E. Impact of Inter-Frame Time Difference

For the purpose of understanding the impact of inter-frame time difference on our proposed models, we train Lite-F2F-M and Lite-O2O-M models by using two more different dataset settings, including i) 5 city cameras arranged along the road and 4 frames sampled from a video, and ii) 5 city cameras arranged along the road and 2 frames sampled from a video. The MAE results are listed in Table XVIII and Table XIX.

In Table XVIII, Lite3_3-F2F-M is the best architecture with the lowest MAE among 9 different architectures of Lite-F2F-M model, and Lite3_1-O2O-M is the best architecture with the lowest MAE among 9 different architectures of Lite-O2O-M model. The results in Table XIX show that Lite3_3-F2F-M has the best performance among 9 different

TABLE XVIII

MAE RESULTS OF OUR END-TO-END METHODS TRAINED BY 5 CAMERAS AND 4 FRAMES IN ONE VIDEO

MAE (Lite-F2F-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	12.747 (Lite1_1-F2F-M)	10.119 (Lite1_2-F2F-M)	15.789 (Lite1_3-F2F-M)
K = 12	8.613 (Lite2_1-F2F-M)	7.324 (Lite2_2-F2F-M)	12.712 (Lite2_3-F2F-M)
K = 16	9.845 (Lite3_1-F2F-M)	9.381 (Lite3_2-F2F-M)	**6.756 (Lite3_3-F2F-M)**
MAE (Lite-O2O-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	11.869 (Lite1_1-O2O-M)	13.57 (Lite1_2-O2O-M)	13.412 (Lite1_3-O2O-M)
K = 12	10.651 (Lite2_1-O2O-M)	13.66 (Lite2_2-O2O-M)	15.863 (Lite2_3-O2O-M)
K = 16	**6.704 (Lite3_1-O2O-M)**	14.966 (Lite3_2-O2O-M)	13.248 (Lite3_3-O2O-M)

TABLE XIX

MAE RESULTS OF OUR END-TO-END METHODS TRAINED BY 5 CAMERAS AND 2 FRAMES IN ONE VIDEO

MAE (Lite-F2F-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	12.747 (Lite1_1-F2F-M)	10.119 (Lite1_2-F2F-M)	15.789 (Lite1_3-F2F-M)
K = 12	8.613 (Lite2_1-F2F-M)	7.324 (Lite2_2-F2F-M)	12.712 (Lite2_3-F2F-M)
K = 16	9.845 (Lite3_1-F2F-M)	9.381 (Lite3_2-F2F-M)	**6.756 (Lite3_3-F2F-M)**
MAE (Lite-O2O-M)			
Kernels \ Layers	L = 1	L = 2	L = 3
K = 8	11.877 (Lite1_1-O2O-M)	14.105 (Lite1_2-O2O-M)	13.983 (Lite1_3-O2O-M)
K = 12	11.047 (Lite2_1-O2O-M)	13.848 (Lite2_2-O2O-M)	15.94 (Lite2_3-O2O-M)
K = 16	**7.717 (Lite3_1-O2O-M)**	15.675 (Lite3_2-O2O-M)	13.506 (Lite3_3-O2O-M)

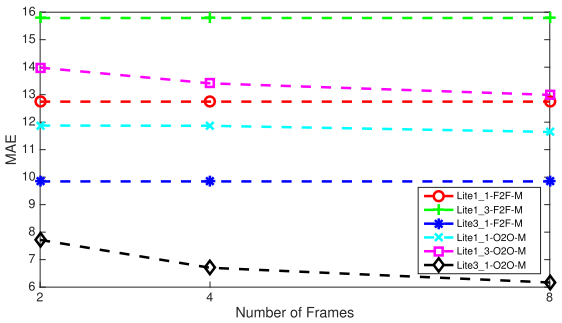


Fig. 8. Impact of inter-frame time difference on MAE.

architectures of Lite-F2F-M model and Lite3_1-O2O-M is the best selection among 9 different architectures of Lite-O2O-M model.

For a clearer illustration, we plot the MAE results of Lite1_1-F2F-M, Lite1_3-F2F-M, Lite3_1-F2F-M, Lite1_1-O2O-M, Lite1_3-O2O-M, and Lite3_1-O2O-M in Fig. 8, which indicates that the influence of inter-frame time difference on Lite-F2F-M is litter, but a smaller inter-frame time difference can help reduce MAE of Lite-O2O-M due to the increase of temporal correlation among frames in each video.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a CityCam-to-Edge cooperative learning framework for vehicle counting. To the best of our knowledge, this is the first work to integrate multiple city cameras with an edge server to accomplish the goal of vehicle counting. Specifically, for the city cameras, we present a lightweight feature extraction scheme to obtain the lightweight FCN by adding multiple convolutional layers with few kernels in FCN, aiming at reducing city cameras' memory consumption and data transmission time. For the edge server, we propose F2F-M and O2O-M models to enhance the counting performance by utilizing the temporal correlation among videos captured from multiple city cameras. Accordingly, two end-to-end vehicle counting models can be obtained, termed Lite-F2F-M and Lite-O2O-M. Through a comprehensive comparison with the state-of-the-art, the advantages of our Lite-F2F-M and Lite-O2O-M can be validated. Especially, Lite-O2O-M model is superior to Lite-F2F-M model in terms of vehicle counting owing to the consideration of the temporal correlation in a video-to-video manner.

In our future work, we will investigate how to reconstruct the traffic videos by using the extracted density maps received from the city cameras with performance guarantee. These reconstructed videos can be further exploited for real-time traffic surveillance, such as vehicle detection, vehicle recognition and vehicle tracking.

REFERENCES

[1] *Traffic Safety Facts Fars/Ges Anuual Report*, NHTSA, Washington, DC, USA, 2013.

[2] *FHWA Forecasts of Vehicle Miles Traveled (VMT)*, U.S. Dept. Transp., Washington, DC, USA, May 2014.

[3] Z. Xiong, Z. Cai, Q. Han, A. Alrawais, and W. Li, "ADGAN: Protect your location privacy in camera data of auto-driving vehicles," *IEEE Trans. Ind. Informat.*, vol. 17, no. 9, pp. 6200–6210, Sep. 2021.

[4] Z. Xiong, H. Xu, W. Li, and Z. Cai, "Multi-source adversarial sample attack on autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2822–2835, Mar. 2021.

[5] J. Wang, Z. Cai, and J. Yu, "Achieving personalized K-anonymity-based content privacy for autonomous vehicles in CPS," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4242–4251, Oct. 2019.

[6] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 766–775, Apr. 2020.

[7] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6492–6499, Dec. 2019.

[8] G. Tesoriere, T. Campisi, A. Canale, A. Severino, and F. Arena, "Modelling and simulation of passenger flow distribution at terminal of Catania airport," *Proc. AIP Conf. Proc.*, vol. 2040, Oct. 2018, Art. no. 140006.

[9] F. Arena and D. Ticali, "The development of autonomous driving vehicles in tomorrow's smart cities mobility," *Proc. AIP Conf. Proc.*, vol. 2040, no. 1, 2018, Art. no. 140007.

[10] B. M. Masini, L. Zuliani, and O. Andrisano, "On the effectiveness of a GPRS based intelligent transportation system in a realistic scenario," in *Proc. IEEE 63rd Veh. Technol. Conf.*, vol. 6, Jan. 2006, pp. 2997–3001.

[11] Y. Riouali, L. Benhlima, and S. Bah, "Extended batches Petri nets based system for road traffic management in WSNs," *J. Sensor Actuator Netw.*, vol. 6, no. 4, p. 30, Dec. 2017.

[12] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, and J. Son, "Digestive neural networks: A novel defense strategy against inference attacks in federated learning," *Comput. Secur.*, vol. 109, Oct. 2021, Art. no. 102378.

[13] A. Toppan, A. Bazzi, P. Toppan, B. M. Masini, and O. Andrisano, "Architecture of a simulation platform for the smart navigation service investigation," in *Proc. IEEE 6th Int. Conf. Wireless Mobile Comput., Netw. Commun.*, Oct. 2010, pp. 548–554.

[14] G. Pau, A. Severino, and A. Canale, "Special issue 'new perspectives in intelligent transportation systems and mobile communications towards a smart cities context,'" *Future Internet*, vol. 11, no. 11, p. 228, 2019.

[15] N. Uchida, S. Takeuchi, T. Ishida, and Y. Shibata, "Mobile traffic accident prevention system based on chronological changes of wireless signals and sensors," *J. Wireless Mob. Netw. Ubiquitous Comput. Dependable Appl.*, vol. 8, no. 3, pp. 57–66, 2017.

[16] D. Biswas, H. Su, C. Wang, J. Blankenship, and A. Stevanovic, "An automatic car counting system using OverFeat framework," *Sensors*, vol. 17, no. 7, p. 1535, Jun. 2017.

[17] V. Astarita, V. P. Giofrè, G. Guido, and A. Vitale, "A single intersection cooperative-competitive paradigm in real time traffic signal settings based on floating car data," *Energies*, vol. 12, no. 3, p. 409, 2019.

[18] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1243–1253, Feb. 2019.

[19] W. Liu, G. Qin, Y. He, and F. Jiang, "Distributed cooperative reinforcement learning-based traffic signal control that integrates V2X networks' dynamic clustering," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 8667–8681, Oct. 2017.

[20] J. S. Pan, I. S. Popa, and C. Borcea, "DIVERT: A distributed vehicular traffic re-routing system for congestion avoidance," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 58–72, Jan. 2016.

[21] H. Hashemi and K. Abdelghany, "End-to-end deep learning methodology for real-time traffic network management," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 10, pp. 849–863, Oct. 2018.

[22] J. Lin, W. Yu, X. Yang, Q. Yang, X. Fu, and W. Zhao, "A real-time en-route route guidance decision scheme for transportation-based cyberphysical systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 3, pp. 2551–2566, Mar. 2017.

[23] S. Gestrelius, A. Peterson, and M. Aronsson, "Timetable quality from the perspective of a railway infrastructure manager in a deregulated market: An interview study with Swedish practitioners," *J. Rail Transp. Planning Manage.*, vol. 15, Sep. 2020, Art. no. 100202.

[24] E. J. Oughton, Z. Frias, S. van der Gaast, and R. van der Berg, "Assessing the capacity, coverage and cost of 5G infrastructure strategies: Analysis of The Netherlands," *Telematics Inf.*, vol. 37, pp. 50–69, Apr. 2019.

[25] R. W. Burchell, D. Listokin, and C. C. Galley, "Smart growth: More than a ghost of urban policy past, less than a bold new horizon," *Housing Policy Debate*, vol. 11, no. 4, pp. 821–879, Jan. 2000.

[26] W. Balid, H. Tafish, and H. H. Refai, "Intelligent vehicle counting and classification sensor for real-time traffic surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1784–1794, Jun. 2018.

[27] N. Wahlstrom, R. Hostettler, F. Gustafsson, and W. Birk, "Classification of driving direction in traffic surveillance using magnetometers," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1405–1418, Aug. 2014.

[28] K. Liu, H. Xiong, and H. He, "New method for detecting traffic information based on anisotropic magnetoresistive technology," *Proc. SPIE*, vol. 8783, Mar. 2013, Art. no. 87830.

[29] A. Daubaras and M. Zilys, "Vehicle detection based on magneto-resistive magnetic field sensor," *Electron. Electr. Eng.*, vol. 118, no. 2, pp. 27–32, Feb. 2012.

[30] Z. Zhang, X. Li, H. Yuan, and F. Yu, "A street parking system using wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 6, 2013, Art. no. 107975.

[31] H. Zhu and F. Yu, "A vehicle parking detection method based on correlation of magnetic signals," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 7, Jul. 2015, Art. no. 361242.

[32] C.-M. Tsai and Z.-M. Yeh, "Intelligent moving objects detection via adaptive frame differencing method," in *Proc. Asian Conf. Intell. Inf. Database Syst.* New York, NY, USA: Springer, 2013, pp. 1–11.

[33] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Statistic and knowledge-based moving object detection in traffic scenes," in *Proc. Intell. Transp. Syst.*, 2000, pp. 27–32.

[34] Y. Zheng and S. Peng, "Model based vehicle localization for urban traffic surveillance using image gradient based matching," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 945–950.

[35] E. Toropov, L. Gui, S. Zhang, S. Kottur, and J. M. Moura, "Traffic flow from a low frame rate city camera," in *Proc. Int. Conf. Image Process.*, 2015, pp. 3802–3806.

[36] K. SuganyaDevi, N. Malmurugan, and R. Sivakumar, "Efficient foreground extraction based on optical flow and smed for road traffic analysis," *Int. J. Cyber-Secur. Digit. Forensics*, vol. 1, no. 3, pp. 177–182, 2012.

[37] Y.-L. Chen, B.-F. Wu, H.-Y. Huang, and C.-J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 2030–2044, May 2011.

[38] Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle detection, tracking and classification in urban traffic," in *Proc. 15th Int. Conf. Intell. Transp. Syst.*, 2012, pp. 951–956.

[39] G. Mo and S. Zhang, "Vehicles detection in traffic flow," in *Proc. 6th Int. Conf. Natural Comput.*, Aug. 2010, pp. 751–754.

[40] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1324–1332.

[41] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "FCN-RLSTM: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 3667–3676.

[42] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 833–841.

[43] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 615–629.

[44] A. Kanungo, A. Sharma, and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," in *Proc. Recent Adv. Eng. Comput. Sci.*, 2014, pp. 1–6.

[45] U. Nagaraj, J. Rathod, P. Patil, S. Thakur, and U. Sharma, "Traffic jam detection using image processing," *Int. J. Eng. Res. Appl.*, vol. 3, no. 2, pp. 1087–1091, 2013.

[46] V. Dangi, A. Parab, K. Pawar, and S. S. Rathod, "Image processing based intelligent traffic controller," *Undergraduate Academic Res. J.*, vol. 6, pp. 13–17, Jul. 2012.

[47] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Understanding traffic density from large-scale web camera data," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5898–5907.

[48] Z. Zhao, H. Li, R. Zhao, and X. Wang, "Crossing-line crowd counting with two-phase deep neural networks," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 712–726.

[49] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 589–597.

[50] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 483–498.

[51] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[52] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[53] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[55] F. Arena, G. Pau, and A. Severino, "A review on IEEE 802.11 p for intelligent transportation systems," *J. Sensor Actuator Netw.*, vol. 9, no. 2, p. 22, Apr. 2020.

Honghui Xu received the bachelor's degree from the University of Electronic Science and Technology of China (UESTC). He is currently pursuing the Ph.D. degree with the Department of Computer Science, Georgia State University (GSU). His research focus on machine learning and deep learning, including the fundamental theory of machine learning, the applications of deep learning in computer vision field, and the topic about privacy-preserving machine learning.

Zhipeng Cai (Senior Member, IEEE) received the B.S. degree from the Beijing Institute of Technology and the M.S. and Ph.D. degrees from the Department of Computing Science, University of Alberta. He is currently an Associate Professor with the Department of Computer Science, Georgia State University, USA. Prior to joining GSU, he was a Research Faculty with the School of Electrical and Computer Engineering, Georgia Institute of Technology. His research areas focus on the Internet of Things, machine learning, cyber-security, privacy, networking, and big data. He was a recipient of an NSF Career Award. He served as the Steering Committee Co-Chair and a Steering Committee Member for WASA and IPCCC. He also served as a Technical Program Committee Member for more than 20 conferences, including INFOCOM, ICDE, and ICDCS. He has been serving as an Associate Editor-in-Chief for *High-Confidence Computing* (HCC) journal (Elsevier), and an Associate Editor for more than ten international journals, including IEEE INTERNET OF THINGS JOURNAL (IoT-J), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (TVT).

Ruinian Li received the Ph.D. degree in computer science from George Washington University in 2018. He is currently an Assistant Professor with the Department of Computer Science, Bowling Green State University (BGSU). His research interests include security and privacy-preserving computations, applied cryptography, and blockchain technology. He has been working in a wide area of social networks, auction systems and the Internet of Things, and his work has been published at top-tier journals, such as *IoT* journal, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Wei Li (Member, IEEE) received the M.S. degree in computer science from the Beijing University of Posts and Telecommunications in 2011 and the Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2016. She is currently an Assistant Professor with the Department of Computer Science, Georgia State University. Her current research spans the area of blockchain technology, security and privacy for the Internet of Things and cyber-physical systems, secure and privacy-aware computing, big data, game theory, and algorithm design and analysis. She won the best paper awards in ACM MobiCom Workshop CRAB 2013 and International Conference WASA 2011, respectively. She is a member of ACM.