Machine Vision and Applications (2021) 32:108
https://doi.org/10.1007/s00138-021-01233-2

ORIGINAL PAPER

ST-CSNN: a novel method for vehicle counting

Kang Yin[1] · Liantao Wang[1] · Jinxia Zhang[2]

Received: 31 December 2020 / Revised: 20 May 2021 / Accepted: 8 July 2021 / Published online: 10 August 2021

Abstract
Vehicle counting using computer vision techniques has potential to alleviate traffic congestion in intelligent transportation system. In this paper, we propose a novel method to count vehicles in a human-like manner. This paper has two main contributions. Firstly, we propose ST-CSNN, which is an efficient, lightweight vehicle counting method. The method counts based on vehicle identity comparison to omit duplicate instances. Combined with the spatio-temporal information between frames, it is able to accelerate speed and improve accuracy of counting. Secondly, we strengthen the method's performance by proposing an improved loss function on the basis of Siamese neural network. Besides, we conduct experiments on several datasets to evaluate the performance of the proposed loss function for verification and the whole method for counting. The experimental results show the practicability of this method for real counting scenes.

Keywords ST-CSNN · Vehicle counting · Siamese neural network

1 Introduction

Vehicles play an important role in life and work, which greatly reduces the time required for people to travel to their destinations. However, as more and more vehicles are put into use, traffic congestion has become a very common phenomenon. Traffic congestion not only affects personal time, but also potentially affects the economy of the entire society [1–5].

In order to alleviate and solve the problem of traffic congestion, first, many physical methods have been proposed, such as increasing road capacity, restricting vehicle flows, reducing road crossings, and vigorously developing public transportation. With the development of computer science and artificial intelligence, people try to combine advanced information, communication, control, and sensing technologies to build a real-time and efficient traffic management and transportation system. Considering the economic cost and sustainability, researchers started with psychology, people's living environment, and driver's habits [6,7] and proposed many methods of traffic flow analysis and statistics. Taking various comprehensive factors into account, they try to propose a method to alleviate or even solve the problem of traffic congestion by formulating a series of driving strategies based on the existing road conditions. A comprehensive study of the traffic flow in certain conditions at certain times is critical to formulating effective traffic strategies and predicting traffic congestion. Thus, a simple and efficient vehicle counting method is especially important.

Vehicle counting method has gone through a long period to develop, which includes several different counting ideas. Detection-based methods [8,9] and motion-based methods [10–12] are first introduced, but are soon abandoned for low detection precision, low frame rate, and lack of motion information. To avoid segmentation of vehicles, holistic methods [13–15] are proposed to process the whole image. With the development of deep learning models, density estimation is popular among recent methods. These meth-

This work was supported in part by the Fundamental Research Funds for the Central Universities (No. B200202213), the Natural Science Foundation of Jiangsu Province (No. BK20201160). The corresponding author is Liantao Wang.

✉ Liantao Wang
ltwang@hhu.edu.cn

Kang Yin
kangyin1997@gmail.com

Jinxia Zhang
jinxiazhang@seu.edu.cn

[1] College of Internet of Things Engineering, Hohai University, Changzhou 213002, China

[2] Key Laboratory of Measurement and Control of CSE, Ministry of Education, School of Automation, Southeast University, Nanjing 210096, China

ods [16–18] can approximate vehicle densities in specific regions, and some of them [19] utilize spatio-temporal information to avoid direct adding counts in frames. Many of them [16–19], however, are too complex in network structures and cost a lot of time in training and application periods.

To overcome the above limitations, we propose a novel vehicle counting method—a highly strengthened spatio-temporal class Siamese neural network (ST-CSNN). In the method, counting is triggered only when new vehicles are detected. Category information is incorporated into the traditional SNN to perform identity comparison. Moreover, the proposed method appropriately utilizes spatio-temporal information in an intuitive manner, and the significance of vehicle motion is dynamically considered, thus lowering the negative influence caused by low frame rates, high occlusion, simply summing the learned densities and so on.

This paper has the following novelties and contributions: (1) We propose the ST-CSNN, which is a lightweight, efficient vehicle counting method. It imitates the manner of human counting and leverages the easiest idea to simplify the process of counting. An adaptive counting strategy is used to dynamically choose vehicles to count; hence, the efficiency of processing whole frame sequences will increase accordingly. Ablation study of our strategy is conducted to demonstrate its effectiveness both on improving counting accuracy and reducing runtime. Besides, compared with LSTM-based work, the proposed idea is much simpler in implementation and application.

(2) A joint loss function is proposed to improve the performance on identification. Original SNN loss performs less accurately when encountered a group of vehicles with only subtle differences. We improve the performance significantly by adding softmax loss. We have conducted experiments on three public datasets to prove the effectiveness of this improvement.

The proposed method summarizes shortcomings from previous methods, and learned advantages from the above. In principle, it is a detection-based method, but it also combines with spatio-temporal information and deep learning technique. In addition, an intuitive counting strategy further strengthens the practicability in real-scene application for tiny network and time costs. We present comprehensive experiments on two datasets covering different conditions of resolution and perspective to show the substantially higher accuracy of ST-CSNN.

We organize the rest of the paper as follows. We first review the relevant works (Sect. 2) and then describe the proposed method and mechanism (Sect. 3). Then, we present and analyze extensive experimental results (Sect. 4), and finally, we outline some conclusions and future directions (Sect. 5).

2 Related work

Vehicle counting methods In 1980s, researchers tried to estimate and control traffic flow by manually counting [20]. Later in [21], manual counting data were evaluated and improved by correcting errors. Even though these data were highly used, it was implicit and requires many corrections. As the need of high accuracy and real-time response, many attempts have been made. Multiple ultrasonic sensors were used to detect obstacles [22], and distributed traffic loop detectors were applied to acquire a large number of measurements spatially distributed over traffic network environment [23]. The above detectors can acquire very precise data in the laboratory, but in real condition which could be rainy, foggy, and dim circumstances, these data are often not reliable even unable to use. Besides, extra costs are required for device installation and further maintenance.

Since traffic cameras have been preinstalled a lot in many high traffic volume areas, video-based vehicle counting becomes popular. With limited number of cameras, correlation information was used to minimize vehicle count estimation errors [24], but this method does not involve visual information. A hierarchical feature-based detector was proposed to work properly both at day and night [25], however, it sacrificed real-time performance. More recently, with the popularity of deep learning, neural network was largely used in vehicle counting [26,27]. In [26], a Faster-RCNN-based system was used to detect vehicles in parking lots, and in [27], a YOLO+KCF model was proposed to address real-time problem, also, an object detection and trajectory monitoring method is proposed in [28]. The above three methods are trajectory detection methods, but all of them detect all vehicles in one frame, which take much computational costs.

Vehicle identity verification In [29], a linear SVM classifier was trained to predict each vehicle object in dense traffic. It alleviated the problem of occlusion to some degree, but the verification accuracy still remained to increase. Using UAV (unmanned aerial vehicle) provided great convenience for large perspective tracking [30–32]. Recently, more and more researchers started working on vehicle re-identification. Liu et al. [33] proposed a coupled triplet loss to further separate positive and negative samples. A GSTE method [34] was proposed by Bai et al. to focus on inner-class performance. Zhou et al. [35] focused on multi-view inference and reconstructed multiple views with only single view input. He et al. [36] presented 3D object retrieval performance with triplet center loss. Combined with generative adversarial network, Lou et al. [37] proposed a FDA net and conducted comprehensive experiments on several challenging datasets.

Siamese neural network It is an important task to compute the semantic similarity between two objects, and Siamese

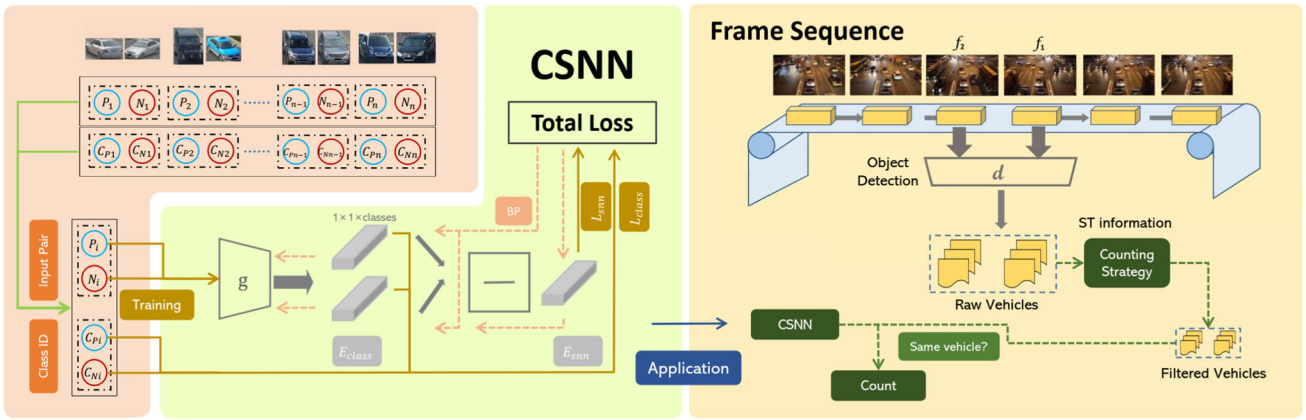


Fig. 1 A comprehensive view of ST-SCNN. It contains three parts which are data preprocessing part (red), training part (green), and application part (blue). In data preprocessing part, vehicles in different classes are paired (P and N represent positive and negative samples, respectively), and each input pair is fed into network g along with its Class ID, which contains the class information of the two vehicles. Then, the network CSNN is mainly used to verify identification, and in addition to the basic loss which measures euclidean distance, we add a softmax loss to precisely distinguish subtle differences between similar vehicles. When applying our method to counting, two consecutive frames are first fed into a object detection network, and unfiltered vehicles are extracted. Then, three counting strategies are taken, in order to get more precise counting results and reduce computational costs (color figure online)

neural network can be the best choice. Ever since it was proposed, it has been applied to various fields, and it has the highest number of applications in image analysis. Proposed by Bromley et al. in the early 1990s [38], Siamese neural network was first designed to solve the problem of signature verification. Later in 2005 [39], LeCun et al. proposed a contrastive energy function for training to distinguish between a genuine pair and an imposter pair, and the similarity metric between pairs was learned directly which could be implicit. To handle this problem, Taigman et al. [40] proposed to add a trainable parameter as a coefficient of the learnt embedding, which was automatically optimized during the training. They also simply used cross-entropy loss as objective function, and it showed great results as well.

Paisios et al. [41] employed Siamese model to recognize similar clothes. Berlemont et al. [42] presented a unique application of Siamese neural network for symbolic gesture recognition. Another interesting topic, using remote sensing technique to do classification, was conducted by Liu et al. [43] and He et al. [44]. It is worth noting that although Siamese neural network has been proposed for many years, there has been a steady stream of applications in recent years, which is enough to demonstrate the value of this model.

3 Methodology

3.1 Outline of ST-CSNN

A comprehensive view of our method is presented in Fig. 1. We basically divide the method into three parts, on the left is a data preprocessing part, where vehicles and their labels are paired in advance. The second part is a training part, where the network learns to verify between vehicles and categorize among classes. Finally, there is a application part. Raw vehicles are adopted from frame sequence, and trained CSNN network and spatio-temporal information are combined to count.

In this figure, P_i and N_i are the paired vehicles and C_{P_i} and C_{N_i} are their class IDs (which could be simply treated as their plates). g is a neural network for feature extraction, and it requires images as the input and outputs a vector E_{class}. E_{class} has the same length as the number of class IDs and could be used to represent the probability that the input vehicle belongs to a certain class. E_{snn} indicates the probability that the two input vehicles belong to the same class. d is another neural network for extracting vehicles and recording their positions. f_1 and f_2 are two consecutive frames in the frame sequence.

In data preprocessing part, the training set is constructed under the following constraints: vehicles are paired and then fed into the network, and paired vehicles from the same classes and from different classes take 50% each; the pair also includes class ID of each vehicle.

In the training part, the image pair will be fed into the network g simultaneously, and two E_{class} embeddings are calculated to get similarity embedding E_{snn}. The similarity loss L_{snn} is acquired by predicting labels with ground truth, and E_{class} and class IDs are used to calculate L_{class}, which is used to categorize vehicles. In the backpropogation, the network is updated by the united L_{snn} and L_{class}. After training, E_{class} functions as an interface for the following part. It is

worth noticing that, during training, the integration of vehicle ID can help SNN learn more discriminative feature embeddings, which can better judge whether two vehicle pictures are the same vehicle; in the test, we make counts by judging whether two vehicles pictures are the same, instead of relying on category prediction. The vehicle to be predicted in the test or real scenarios does not have to be the vehicle in the training set, and the test set used in the experiment also contains vehicles (license plate) that are not included in the training set. Adding such class ID information is to improve the recognition ability of subtle differences between similar vehicles, instead of giving every vehicle a unique coding. After training, structure of the whole network is fixed, and thus, the class ID vector will not be that much big as unacceptable.

In application part, consecutive frames are presented in a sequence. Two frames are processed a time, where we denote them by f_1 and f_2 here. At first, f_1 and f_2 are fed into network d to detect all existing vehicles. Because there are many mature object detection methods, we simply use a toolbox Detectron2 [56] here. To count, we only consider newly appeared vehicles in f_2 compared with those in f_1. Based on spatio-temporal information, some vehicles in f_2 could be directly added into the count, and the rest are selectively paired and fed into network g, which will then output whether the vehicles in a pair is the same and count.

Combining the spatio-temporal information between frames, the human-like counting strategies used for this specific task can be concluded as follows:

– *Only count big vehicles* We tend to focus on big things when we start to count because it is easier to count the number of them. In this method, we fix our model's attention on big vehicles, as vehicle stream is flowing, and finally, everyone in it will be taken into account when getting closer to the camera. Specifically, we will screen the detected vehicles with two ideas. The first idea is that, given an image, only vehicles whose area exceeds the set threshold t_1 will be retained. As for the second idea, we sort the detected vehicles in descending order of area and take the vehicle with the largest area as the benchmark. Similarly, set a threshold t_2 so that only vehicles with an area exceeding the benchmark a certain percentage are retained. The vehicles that satisfy either strategy one or two are the final detected vehicles in an image. Denote set of the original detected vehicles by V, the formulas are defined as:

$$N_1 = \left\{ V_i \mid V_i \in V, area\left(\frac{V_i}{image}\right) > t_1 \right\}, \quad (1)$$

$$N_2 = \left\{ V_i \mid V_i \in V, area\left(\frac{V_i}{V_{max}}\right) > t_2 \right\}, \quad (2)$$

$$N = N_1 \cup N_2, \quad (3)$$

where N_1 and N_2 represent detected vehicles after strategy one and two, respectively, and N is the total of the two.

This strategy makes our method insensitive to different shooting views, whether it is high-altitude long-distance shooting or low-angle close-up shooting, we can greatly reduce the number of vehicles detected on an image.

– *Only compare adjacent vehicles* When comparing vehicles in two consecutive frames, we do not take two separate vehicle as the same one. That is the other principle we set for our method. In practice, position of all vehicles is saved during detection period; once two vehicles positions exceed the limit, they will be automatically viewed as different vehicles. To implement this idea, we construct a circle on the center coordinate of the vehicle for each vehicle based on the coordinate information. The radius is proportional to the area of the vehicle. According to the traffic volume on the road, the radius will be adjusted adaptively. Only when two circles in different frames intersect will the two vehicles be compared. As shown in Fig. 2, black dots are vehicles, and those who are enclosed by dash orange circles are vehicles detected in frame f_1, and those enclosed by green circles are detected vehicles in frame f_2. We present an adaptive adjustment for the selection of radius. When the road suffers high traffic volume, we set a small value (r_1) for radius while we set a relative big value (r_2) for it when there are few vehicles on the road. In the two examples given in Fig. 2, we greatly reduce unnecessary comparisons, and we only need to compare 9 and 2 times under two different conditions, which without the strategy should be 49 and 4 times, respectively. Specific implementation is illustrated in Sect. 4.5.

This strategy largely avoids comparing vehicles on different lanes, thereby improving the accuracy of recognition and operating efficiency.

– *Avoid counting all frames* In many underdeveloped areas, equipment cannot be effectively updated. Although high frame rate cameras have been used in many areas today, there are still low quality cameras (1fps or even worse) used in many areas. In order to show the universality of our model, we manually reduce frame rate to 1fps. On the one hand, reducing the frame rate can simulate the actual performance of our method in underdeveloped areas, and on the other hand, it can alleviate the pressure of the real-time video transmission stream and ensure the timeliness of data analysis while ensuring the accuracy of counting.

The above counting strategies screen most vehicles, which drastically improved the efficiency in the counting. Traditional vehicle counting methods usually require a density map to depict the density of pedestrians and vehicles in an image, so as to approximate the number. Such kind of meth-

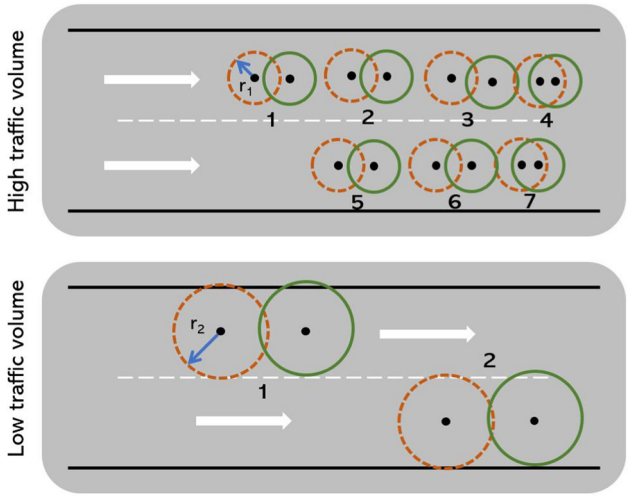


Fig. 2 Only adjacent vehicles, which have overlapping areas, are compared. Adaptive adjustment is made given different traffic conditions

ods are complex and require much storage and computation resources; in contrast with that, our vehicle counting method is much more intuitive by imitating the behavior of human counting, which is making count when seeing a vehicle different from the previous ones in the scene. Algorithm 1 and algorithm 2 outline the training and application process of our method. In algorithm 1, S is the whole training set, where S^* stands for the $*$ th vehicle class in S and S^i_* stands for the i th vehicle image in class S^*. E and L represent embedding and loss function, respectively. In algorithm 2, f_1 and f_2 are two consecutive frames that are being processed. V^* and C^* denote the vehicles left after strategy process. $Strategy_1$ and $Strategy_2$ correspond to strategies mentioned above.

Algorithm 1: Training

Input: Paired Image Set:$\{(S^1_m, S^1_n),\ (S^2_m, S^{\neg 2}_n), \ldots, (S^i_m, S^i_n), (S^j_m, S^{\neg j}_n) \mid S^i_* \in S^i, S^* \in S\}$. Initialized parameters θ in layers of network g, training epoch t, learning rate η, batch size k.
Output: Network parameters θ

for $i \leftarrow 1$ *to* t **do**
  for *each mini-batch* **do**
    $E_{snn}, E_{class} \leftarrow g((s,s)_{[1:k]^T}, \theta^i)$
    $L_{snn} \leftarrow \|E_{snn}\|_2$
    $L_{class} \leftarrow softmax(E_{class}, class\ ID)$
  end
  $L^i \leftarrow \sum(\alpha L_{snn} + (1-\alpha)L_{class})$
  $\theta^{i+1} \leftarrow \theta^i - \eta \frac{\partial L^i}{\partial g(x^i)} \cdot \frac{\partial g(x^i)}{\partial \theta^i}$
end

Algorithm 2: Application

Input: Frame Sequence I:$\{I_1, I_2, \ldots, I_n\}$, $I_j(k)$ is the k^{th} vehicle in frame j. Trained network $g(x)$. Detection network $d(x)$. Denote E_{class} by E, vehicles detected in f_1 and f_2 by V_1 and V_2.
Output: Vehicle counting result C.

Initialize: $f_1 \leftarrow I.pop$
  $V_1 \leftarrow d(f_1)$
  $V_1^* \leftarrow$Strategy$_1(V_1)$
  $C \leftarrow V_1^*$
while *not EOFrame* **do**
  $f_2 \leftarrow I.pop$
  $V_2 \leftarrow d(f_2)$
  $V_2^* \leftarrow$Strategy$_1(V_2)$
  $E_1, E_2 \leftarrow g(V_1^*), g(V_2^*)$
  $C^* \leftarrow$Strategy$_2(\|E_1, E_2\|_2)$
  $C \leftarrow C + C^*$
  $f_1 \leftarrow f_2,\ V_1^* \leftarrow V_2^*$
end

3.2 Proposed loss function

The structure of CSNN with proposed loss function is shown in Fig. 3. As mentioned before, we add class loss L_{class} to get better performance on the condition that there are many similar vehicles belonging to different classes. Although SNN loss alone is capable of extracting enough meaningful embeddings for vehicle verification tasks, it is not sensitive with very similar vehicles. Softmax loss is widely adopted in image classification tasks, and it is especially suitable for this unique task. In the training process, vehicles are grouped by their IDs, whether they share the same brand, color, and appearance. However, vehicles share the above characteristics only have subtle differences in detail, and it is almost impossible for simple SNN loss to recognize among them all, while softmax loss focuses on vehicle IDs, which means similar vehicles could be distinguished with a much larger possibility. As for softmax layer, each neuron will be normalized:

$$f(n_i) = \frac{e^{n_i}}{\sum_j e^{n_j}}, \tag{4}$$

where n_i represents the ith neuron in the layer. Thus, the softmax loss corresponding to the class c can be denoted as:

$$l = -log\left(\frac{e^{n_c}}{\sum_j e^{n_j}}\right) = -n_c + log\left(\sum_j e^{n_j}\right). \tag{5}$$

For the batch of k input pairs, the whole loss is defined as:

$$L_{class} = \frac{\sum_k (-n_c^k + log(\sum_j e^{n_j^k}))}{k} \times 2. \tag{6}$$

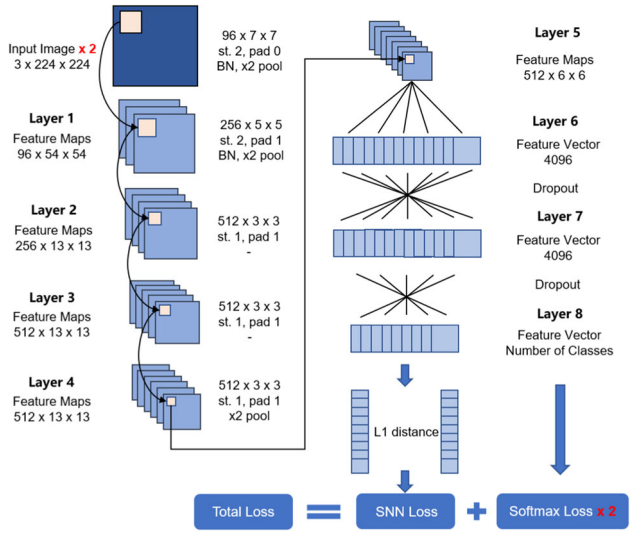


Fig. 3 A high-performance model for verification. Only one branch of the network is depicted

For verifying identity between an input pair, each one of the paired vehicles is separately fed into the network, where $h_{1,l}$ represents embedding vector of the lth layer for the first image and $h_{2,l}$ represents embedding vector of the l^{th} layer for the second image. After several hidden layers, two highly representative embeddings are acquired in the $L-1th$ layer, and the last fully connected layer functions as a coefficient of L2 distance calculated by the pair. Therefore, the final output $\hat{y}$ is deduced:

$$\hat{y} = \sigma\left(\sum_i \beta_i \| h^i_{1,L-1}, h^i_{2,L-1} \|_2\right), \tag{7}$$

where β represents weight vector in the last layer and $\sigma(\cdot)$ represents sigmoid units which map the output into (0, 1) for prediction.

Since our task mainly focusses on whether two objects are from the same target, it is naturally linked with binary classification, so that we use binary cross-entropy loss in our model. Given an image pair (p, n) in which p and n could come from the same class or different classes, its ground truth label is denoted by y and the result predicted by model is denoted by $\hat{y}$. Thus, in the pair level, the SNN loss function is as follows:

$$l(p,n) = y(p,n)\log\hat{y}(p,n) + (1 - y(p,n))\log\left(1 - \hat{y}(p,n)\right). \tag{8}$$

For the batch of k input pairs, the whole loss is defined as:

$$L_{snn} = \frac{\sum_{i=1}^{k}\left(y_i \log\hat{y}_i + (1-y_i)\log\left(1-\hat{y}_i\right)\right)}{k}. \tag{9}$$

Finally, by adding a hyperparameter α to adjust between these two loss functions, our loss function is formulated as follows:

$$L = \alpha L_{snn} + (1-\alpha)L_{class}. \tag{10}$$

4 Experiment

4.1 Hardware platform

In the experiment below, we only use a basic PC to run. The information of the device is: CPU AMD Ryzen 5 3600, Memory 32G DDR4, GPU GTX 1080Ti 11G x1.

4.2 Datasets and protocols

Verifying vehicle's identity is a challenging work, given the condition that it varies in color, size, shape, brand, and so on. The scale in single existing vehicle datasets is far from enough for explicit verification, because single dataset only includes limited meaningful vehicle information. However, vehicles in urban and rural areas often differ a lot in shape, category, and brand. Training and testing on single dataset are not persuasive. Hence, we train our model on three famous datasets (VERI-Wild, Vehicle-ID, and VeRI-776) separately. The code for the whole experiment is uploaded on GitHub (link: github.com/Kang1121/ST-CSNN).

(1) *VERI-Wild* We use VERI-Wild Dataset [37] as our first choice to train the network. This dataset uses a camera network consisting of 174 cameras distributed in a urban district and continuously captures vehicles for a month. With altogether 40671 vehicles (416314 images), VERI-Wild has the following advantages compared with existing vehicle datasets: a. complex capture conditions; b. large time span involving severe illumination and weather changes; c. unconstrained capture conditions in the wild.

(2) *Vehicle-ID* The VehicleID dataset [33] contains data captured during daytime by multiple real-world surveillance cameras distributed in a small city in China. There are 26267 vehicles (221763 images in total) in the entire dataset. Each image is attached with an id label corresponding to its identity in real world.

(3) *VeRI-776* This dataset [45] contains over 50k images of 776 vehicles captured by 20 cameras covering an 1.0 km^2 area in 24 h; although the number of vehicles is not quiet satisfactory, it provides abundant semantic information as well as spatio-temporal information.

(4) *UA-DETRAC* This dataset [46] is a challenging real-world multi-object detection and multi-object tracking benchmark. The dataset consists of 10 hours of videos

at 24 different locations at Beijing and Tianjin in China. The original videos are recorded at 25 fps, with resolution of 960 x 540 pixels. It contains many tricky scenes like highly congested crossroads, high-speed driving, and high occlusion.

(5) *Real-scene Cameras* This dataset is from surveillance cameras at intersections on a city road in Zhejiang province. There are four cameras, with a total length of eight hours' video. Altogether 29,374 frames are included and the frame rate is 1 fps. The size of the original pictures is 1600 x 1200 and 1920 x 1080. This dataset is of great practical significance, because it most truly reflects the vehicle flow under domestic traffic conditions, which includes many common conditions like high congestion, high-speed driving, and high occlusion.

To fairly judge the performance of our model, two metrics—mean average precision (mAP) and cumulative match curve (CMC)—are introduced.

mAP is used when the gallery set contains multiple corresponding results given a specific query. Usually, average precision (AP) is calculated by:

$$AP = \frac{\sum_{k=1}^{n} P(k) \times gt(k)}{N_{gt}}, \tag{11}$$

where n represents the number of results retrieved from the gallery set, $P(k)$ indicates the precision from result 1 to result k, and $gt(k)$ stands for the ground truth of the kth result, and N_{gt} represents the ground truth number of results. Accordingly, mAP metric is formulated as follows:

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}, \tag{12}$$

where Q is the number of queries in the query set.

CMC is used when the gallery set contains only one corresponding result given a specific query. Hence, the formula of CMC metric is as follows:

$$CMC_k = \frac{\sum_{q=1}^{Q} gt(q,k)}{Q}, \tag{13}$$

where $gt(q,k) \in \{0, 1\}$ and equals 1 if and only if ground truth result is among top k retrieved results.

The MAE and the MSE are used for evaluation in counting which are defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}|, \tag{14}$$

$$MSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}|^2}, \tag{15}$$

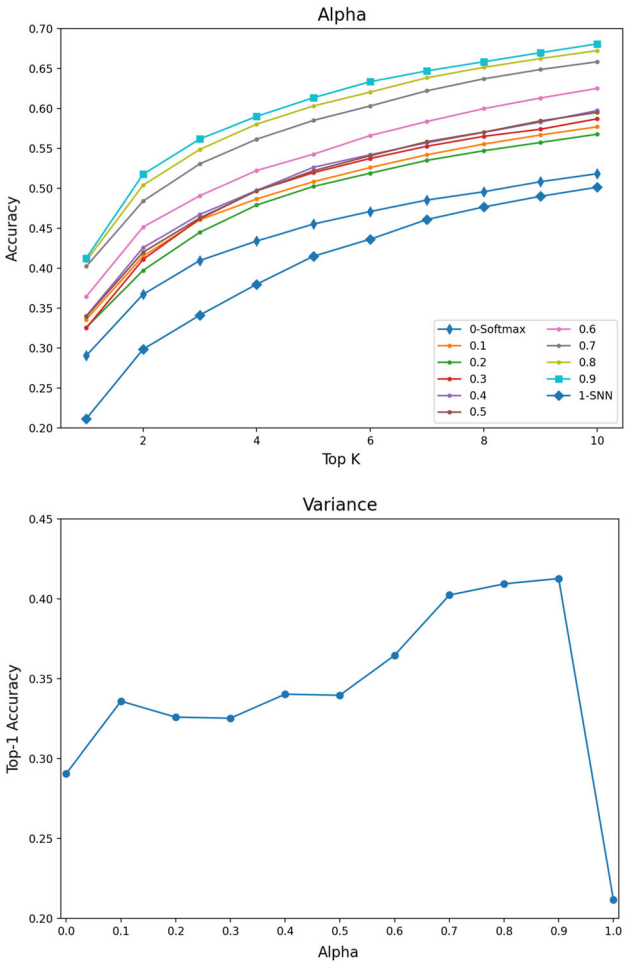


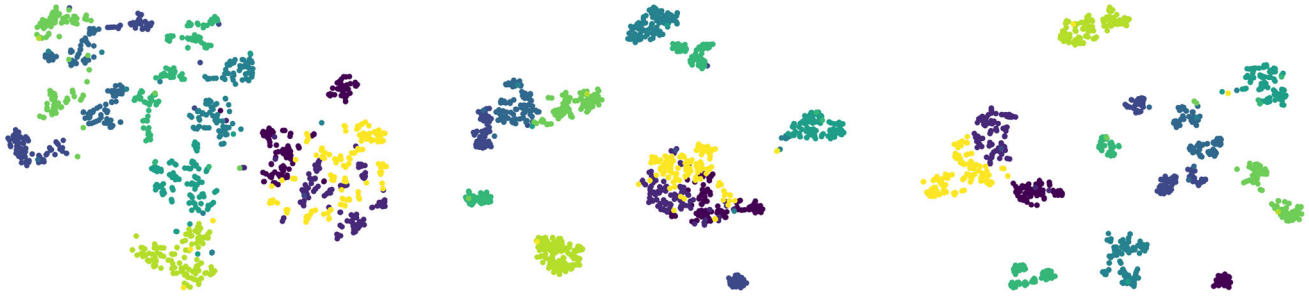
Fig. 4 CMC performance comparison by adjusting the hyperparameter α

where N is the number of images in one test sequence and C_i^{GT} is the ground truth of counting.

4.3 Evaluation of the proposed loss function

In Sect.3.2, the hyper-parameter α is used to balance between loss functions. In order to select the best α for the model, we randomly sample a subset of the training set VERI-Wild, and for simplicity, we only experiment on this dataset. Totally, 1000 vehicles (9524 images) are selected to form the tiny training set. For the same reason, we only train on our baseline VGGM. To reach a relatively precise value, fixing other parameters, we divide the range of α from 0 to 1 by ten. After around 180 epochs, the model converges and finds the best performance result around 0.9.

We test our model on standard small test set (3000 vehicles) provided by the author. It is a remarkable fact that simply using softmax loss or SNN loss cannot get satisfied results, while an appropriate combination is relatively effective. As shown in Fig. 4, the relation between α and Top-K accuracy



(a) SNN Loss **(b)** Softmax Loss **(c)** Joint Loss

Fig. 5 Visualization of classification for ten vehicle IDs with three loss functions

is very clear. At first, α is 0, which means only softmax loss works, and the whole accuracy remains in a low position. When α starts to increase, Top-K accuracy increases accordingly at a rapid race, then slows down its pace, and gradually reaches its peak performance when α is 0.9. Finally, when α is 1, which means only SNN loss functions on the model, the performance turns out to be bad as well. Same experiments are also conducted on other two datasets, and it shows quite similar results. In the following experiments, we set the value of α as 0.9 by default.

To further demonstrate the effectiveness of additive softmax loss, we visualize feature embeddings by reducing their dimensions in 2D (Fig. 5). Different colors represent different vehicle IDs, and 10 vehicles (1100 images) are randomly sampled in total. Comparing (a) and (b), it is obvious to see that the learned representation by softmax loss can better separate vehicles and group them by their IDs in feature space than SNN loss. However, single softmax loss cannot well separate some different classes of vehicles, while joint loss function can effectively distinguish vehicles by their IDs. It is important to note, in (b) and (c), some vehicles in the same color are separated apart by two groups that is because both front and rear perspective images of vehicles are in the dataset.

4.4 Comparison with state-of-the-art methods

To verify the proposed model is compatible and can be used on complex scenarios, we test its performance on three commonly used vehicle re-identification datasets mentioned above. The sample selection for training and testing all follows the instructions in the datasets; the size of test set is 10k, 2.4k, and 0.2k, respectively. He initialization [47] is used to initialize the model, and Adam optimizer [48] is set by default. Learning rate starts from 0.0001 and is divided by 2 every 25 epochs (one forward and backward pass of all the training examples). The model is going to stop after 100 epochs. The size of mini-batch is set to 80. In [57], local response normalization is used in the first two hidden layers to enhance the generalization ability of model by inhibiting neurons with smaller feedback; here we replace it with batch normalization [25] for better model performance. All the experiments are based on PyTorch. The data and methods for comparison are from experiments in [37].

Table 1 Performance on the VERI-Wild dataset

Methods	mAP	Top-1	Top-5
GoogLeNet [49]	**21.53**	**44.61**	63.55
Triplet [50]	9.93	33.46	51.36
Softmax [45]	17.62	37.94	59.89
CCL [33]	14.81	44.6	60.95
HDC [51]	18.30	43.97	64.89
SNN+Softmax (Ours)	18.52	43.73	**65.89**

Bold represents the number of best performance

4.4.1 Quantitative evaluations on VERI-Wild

Table 1 presents performance comparisons on VERI-Wild dataset. The results show that the proposed loss generally does better than most methods on mAP and Top-1 performance. Besides, although our loss function is worse than GoogLeNet [49] in the Top-1 match rate and mAP, it achieves a better performance on the top 5 match rate, implying better recall capability benefiting from the softmax loss. It is worth noting that GoogLeNet [49] and HDC [51] have deeper network structures, and we achieve 1.2% and 1.5% improvements over HDC [51] on mAP and Top-5 performance. We should point out that most previous work and ours are based on the backbone of simple CNN models (e.g., VGG-M or VGG-9), while GoogLeNet has a much deeper and more complicated structure. We can also observe that our loss has outperformed CCL [33] with significant mAP gain of 3.71%.

Table 2 Performance on the VehicleID dataset

Methods	mAP	Top-1	Top-5
LOMO [52]	–	15.32	25.29
DGD [53]	–	37.33	57.82
GoogLeNet [49]	40.39	38.27	59.39
FACT [45]	–	39.92	60.32
XVGAN [54]	–	44.89	66.65
CCL [33]	38.6	35.68	56.24
Mixed Diff [33]	45.5	41.05	63.38
SNN+Softmax (Ours)	**51.06**	**45.48**	**67.35**

Bold represents the number of best performance

Table 3 Performance on the VeRI-776 dataset

Methods	mAP	Top-1	Top-5
LOMO [52]	9.64	25.33	46.48
DGD [53]	17.92	50.70	67.52
GoogLeNet [49]	17.81	52.12	66.79
FACT [45]	18.73	51.85	67.16
XVGAN [54]	24.65	60.20	77.03
SiameseVisual [55]	29.48	41.12	60.31
FACT+Plate+STR [45]	27.77	61.44	78.78
SNN+Softmax (Ours)	**44.5**	**77.28**	**88.52**

Bold represents the number of best performance

4.4.2 Quantitative evaluations on VehicleID

The results on VehicleID are shown in Table 2. We further compare our method with LOMO (using texture features) [52], DGD (domain guided dropout) [53], FACT (model fusion) [45], XVGAN (using GAN to improve performance and focus on multi-view learning) [54], and Mixed Diff (model fusion of CCL) [33].

The experimental results show that our network performance significantly outperforms the GoogleNet which is much deeper than VGGM. Moreover, we can observe that treating each vehicle ID as the granularity of categorization rather than each vehicle model, the mAP, Top-1, and Top-5 performance reaches up to 5.56%, 4.43%, and 3.97%, respectively, when compared to Mixed Diff [33]. In particular, our method even performs better than XVGAN [54], which should have to perform best among the methods. Such significant improvements can be attributed to two aspects. First, the additive softmax layer focuses on vehicle IDs, which results in more distances of latent embeddings. Second, the proposed model is simple enough to work without much data.

4.4.3 Quantitative evaluations on VeRI-776

Table 3 lists the results on VeRI-776 dataset. It's obvious that our method outperforms all provided methods with a much superior performance. Compared to SiameseVisual [55], the incremental improvements on our method demonstrate the class information among different vehicles can be significantly utilized, while the spatio-temporal information with LSTM is plain in verification tasks. More specifically, SNN+Softmax loss outperforms FACT+Plate+STR [45], indicating effectiveness of holistic information rather than only plate regions.

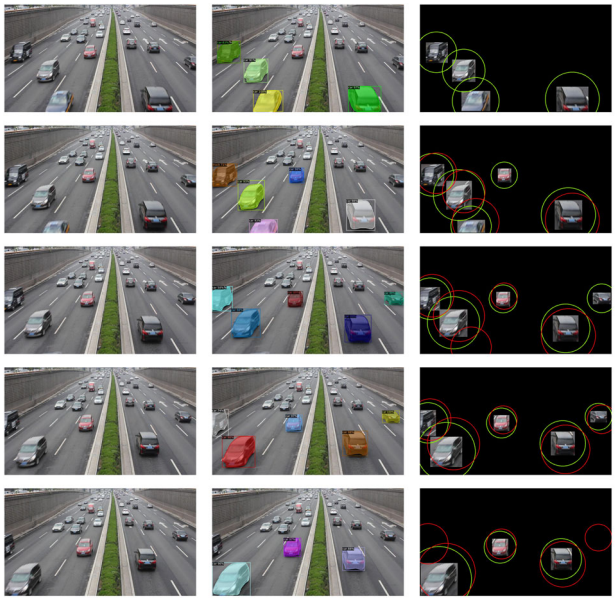
Fig. 6 Some examples from UA-DETRAC dataset

4.5 Evaluation on UA-DETRAC

We first use toolbox Detectron2 [56] to extract vehicles (Fig. 6), and for precise detection, we use the pretrained model *faster-rcnn-X-101-32x8d-FPN-3x* in the model zoo. Shown in Fig. 7, five consecutive frames are selected for a simplest explanation of counting. Notice that we won't put all the vehicles into the model, but calculate the area of bounding box and sort them in descending order. A threshold is set, in this experiment it is 0.25, to make sure, only those whose areas are greater than the threshold times the biggest area can be taken into account. In this example, although over 30 vehicles appear in this frame, at most 5 vehicles conform to the specification, which reduces the cost to a great extent.

We do so to reduce the calculation because those vehicles that driving far away have been counted in previous frames, and also to increase verification accuracy because of low-resolution result in difficult feature extraction.

The location of vehicles is stored beforehand, and comparison only happens when distance between the two is close enough. According to different road conditions, we have different definitions of "close." When the traffic stream flows slowly, a tight constraint on distance is not required; however, it is necessary to compare between distant vehicles when they run fast. To illustrate, each vehicle is enclosed by a green circle, which is its "territory," and red circle represents vehicles' territories in last frame. Only when red circle intersects green circle will two vehicles be compared. Hence, the number of four comparisons is 8, 7, 7, and 4, respectively. We set differ-



(a) Original frames **(b)** Labeled frames **(c)** Extracted vehicles

Fig. 7 A simple example to illustrate how to optimize counting procedure

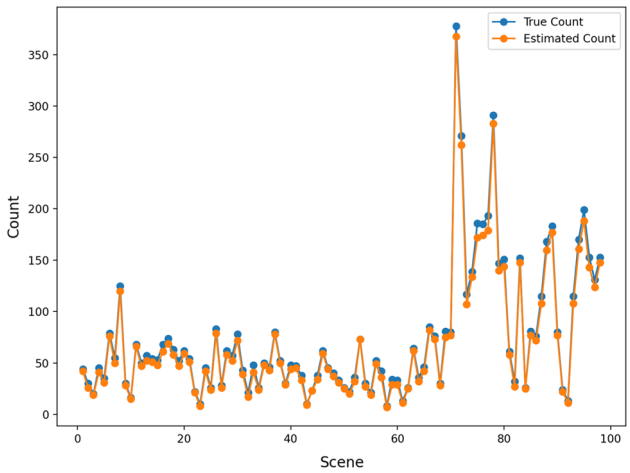


Fig. 8 Counting result on UA-DETRAC dataset

ent radius coefficients for congested and open roads, which are 1.3 and 1.

We evaluate on both training set and test set of this dataset, which altogether has 100 different scenes. We manually counted the number of vehicles in each scene as the true count. The counting result is shown in Fig. 8. The MAE and MSE result for the whole test set is 3.88 and 44.12, respectively. In some cases, our method might miss some vehicles. Shown in Fig. 9, we can see that a taxi in the left lane is occluded by a black van from beginning to end. Although we can manually count this vehicle, our model cannot tell whether it is a vehicle, because most of its body is invisible. This is a common problem for video-based counting

(a) Vehicle occlsion **(b)** Obstacle occlsion

Fig. 9 Two typical miss counts

Table 4 Comparison between optimized and original counting strategy

	Time	MAE	MSE
W/ optim	0.95	3.23	18.44
W/O optim	2.24	4.18	26.73

The unit of column time is hour (s)

methods. On the condition that objects are captured from a oblique perspective, this error seems hard to eliminate without auxiliary support like multi-cameras. This kind of problem also happens in video ID 40991 and 40992, where a camera is positioned behind a big pole, in which case our method becomes unreliable. After removing these two abnormal scenes, the final MAE and MSE result comes to 3.23 and 18.44. In many other scenes, vehicles are running in a high speed, which results in very blurry images, and consequently lower counting accuracy.

To verify the effectiveness of our optimization for the counting strategy, we also conduct comparison experiment. Presented in Table 4, in addition to less time cost, optimized counting strategy also achieves higher accuracy, as redundant comparisons will inevitably result in miss count. Finally, the optimized counting method proceeds about 4 frames per second.

4.6 Evaluation on real-scene cameras

This data set has a total of 14 video clips (Fig. 10), each of which is 30 minutes long. There are a total of four different scenes, with two cameras having a viewing angle parallel to the lane and the other two cameras shooting at an oblique

Fig. 10 Examples in real scenes

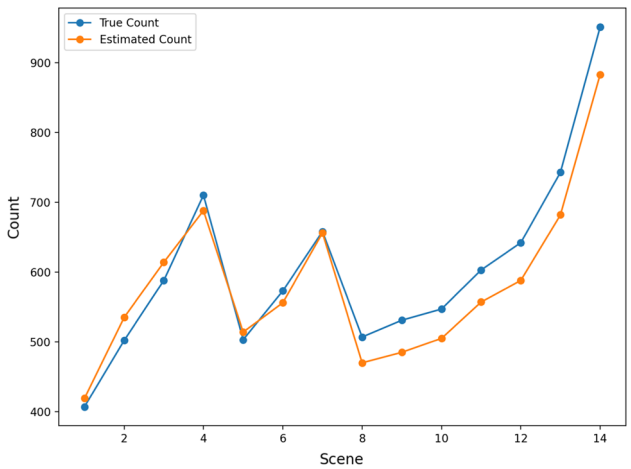


Fig. 11 Counting result on real-scene cameras dataset

angle. We use the same process as Sect. 4.5 to evaluate on this dataset.

The experimental result is shown in Fig. 11. The first seven scenes were shot at parallel angles, while the last seven scenes were shot at oblique angles. It is obvious from the figure the importance of the angle of view to the counting method. Considering the actual situation, because there are bus lanes on this road section, when shooting from an oblique angle of view, there is a high possibility that the smaller vehicles will be completely blocked by the bus, resulting in miss counting.

But even so, our method still performs well on this dataset, especially when shooting in a direction parallel to the road. The MAE and MSE result is 34.07 and 1530.92, respectively. Please note that the base of our vehicle counts is very large, and the average number of vehicles per scene is 600. The total accuracy of 14 scenes (8152 vehicles estimated/8465 vehicles ground truth) is up to 96.3%.

5 Conclusion & future work

In this paper, we propose a novel vehicle counting method ST-CSNN. It counts in a human counting manner, by utilizing spatio-temporal information between frames. A proposed loss function is used to improve the verification performance. The effectiveness of the proposed loss function and the whole method is evaluated on altogether five datasets, and all shows great performance. Besides, it is a lightweight method for training and application.

Despite the advantages, we still have the problem of miss count when encountered high occlusion conditions, and we do not evaluate on many vehicle counting dataset because we require consecutive frames in the dataset. We will then focus on improving the accurate by solving the problem of high occlusion, and we will improve our method and conduct further experiments on more datasets to show our method's robustness and practicability.

Declaration

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Kong, X., et al.: Urban traffic congestion estimation and prediction based on floating car trajectory data. Future Gener. Comput. Syst. **61**, 97–107 (2016)
2. Zheng, Y., et al.: Urban computing: concepts, methodologies, and applications. ACM Trans. Intell. Syst. Technol. (TIST) **5**(3), 1–55 (2014)
3. Sweet, M.: Does traffic congestion slow the economy? J. Plan. Lit. **26**(4), 391–404 (2011)
4. Zhang, J., Ioannou, P.A.: Longitudinal control of heavy trucks in mixed traffic: environmental and fuel economy considerations. IEEE Trans. Intell. Transp. Syst. **7**(1), 92–104 (2006)
5. Alnawaiseh, N.A., Hashim, J.H., Md Isa, Z.: Relationship between vehicle count and particulate air pollution in Amman, Jordan. Asia Pac. J. Public Health **27**(2), NP1742–NP1751 (2015)
6. Zhou, T., Guo, Y., Yang, Y., et al.: The School District Reorganization by Combining with Traffic Congestion Data[C]//2019 International Conference on Computer, Information and Telecommunication Systems (CITS). IEEE, 2019: 1-5. [] Adebisi, Olusegun. "Improving manual counts of turning traffic volumes at road junctions." Journal of transportation engineering 113.3 (1987): 256-267
7. Sun, D., Zhang, C., Zhang, L., et al.: Urban travel behavior analyses and route prediction based on floating car data. Transp. Lett. **6**(3), 118–125 (2014)
8. Zheng, Y., Silong P.: Model based vehicle localization for urban traffic surveillance using image gradient based matching. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems. IEEE, (2012)
9. Toropov, E. et al.: Traffic flow from a low frame rate city camera. In: 2015 IEEE International Conference on Image Processing (ICIP). IEEE, (2015)

10. Chen, Y.-L., et al.: A real-time vision system for nighttime vehicle detection and traffic surveillance. IEEE Trans. Ind. Electron. **58**(5), 2030–2044 (2010)
11. Chen, Z., Ellis, T., Velastin, S.A.: Vehicle detection, tracking and classification in urban traffic. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems. IEEE, (2012)
12. Mo, G., Sanyuan, Z.: Vehicles detection in traffic flow. In: 2010 Sixth International Conference on Natural Computation. Vol. 2. IEEE, (2010)
13. Xia, F., Shanghang, Z.: Block-coordinate frank-wolfe optimization for counting objects in images. In: Advances in neural information processing systems workshops. **2**,(2016)
14. Gonçalves, W.N., Machado, B.B., Bruno, O.M.: Spatiotemporal Gabor filters: a new method for dynamic texture recognition. arXiv preprint arXiv:1201.3612 (2012)
15. Lempitsky, V., Zisserman, A.: Learning to count objects in images. Adv. Neural Inform. Process. Syst. **23**, 1324–1332 (2010)
16. Onoro-Rubio, D., Roberto, J.L.-S.: Towards perspective-free object counting with deep learning. In: European Conference on Computer Vision. Springer, Cham, (2016)
17. Arteta, C., Lempitsky, V., Zisserman, A.: Counting in the wild. In: European conference on computer vision, Springer, Cham (2016)
18. Zhang, S. et al.: Understanding traffic density from large-scale web camera data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017)
19. Zhang, S. et al.: Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In: Proceedings of the IEEE international conference on computer vision. (2017)
20. Adebisi, O.: Improving manual counts of turning traffic volumes at road junctions. J. Transp. Engi. **113**(3), 256–267 (1987)
21. Zheng, P., Mike, M.D.: An investigation on the manual traffic count accuracy. Procedia Soc. Behav. Sci. **43**, 226–231 (2012)
22. Agarwal, V., Murali, N.V., Chandramouli, C.: A cost-effective ultrasonic sensor-based driver-assistance system for congested traffic conditions. IEEE Trans. Intell. Transp. Syst. **10**(3), 486–498 (2009)
23. Ramezani, A., Behzad, M.: The traffic condition likelihood extraction using incomplete observation in distributed traffic loop detectors. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, (2011)
24. Zhuang, P., Shang, Y., Hua, B.: Statistical methods to estimate vehicle count using traffic cameras. Multidimens. Syst. Signal Process. **20**(2), 121–133 (2009)
25. Robert, K.: Video-based traffic monitoring at day and night vehicle features detection tracking. In: 2009 12th International IEEE Conference on Intelligent Transportation Systems. IEEE, (2009)
26. Khan, G. et al.: Deep-Learning Based Vehicle Count and Free Parking Slot Detection System. In: 2019 22nd International Multitopic Conference (INMIC). IEEE, (2019)
27. Liang, H., et al.: Vehicle Counting System using Deep Learning and Multi-Object Tracking Methods. Transp. Res. Rec. **2674**, 114–128 (2020)
28. Bui, N., Yi, H., Cho, J.: A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 578-579) (2020)
29. Van Pham, H., Byung-Ryong, L.: Front-view car detection and counting with occlusion in dense traffic flow. Int. J. Control Autom. Syst. **13**(5), 1150–1160 (2015)
30. Xiang, X., et al.: Vehicle counting based on vehicle detection and tracking from aerial videos. Sensors **18**(8), 2560 (2018)
31. Cazzato, D., Cimarelli, C., Sanchez-Lopez, J.L., Voos, H., Leo, M.: A survey of computer vision methods for 2D Object detection from unmanned aerial vehicles. J. Imaging **6**(8), 78 (2020)
32. Khan, N.A., Jhanjhi, N.Z., Brohi, S.N., Usmani, R.S.A., Nayyar, A.: Smart traffic monitoring system using unmanned aerial vehicles (UAVs). Comput. Commun. **157**, 434–443 (2020)
33. Liu, H. et al.: Deep relative distance learning: Tell the difference between similar vehicles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016)
34. Bai, Y., et al.: Group-sensitive triplet embedding for vehicle reidentification. IEEE Trans. Multimed. **20**(9), 2385–2399 (2018)
35. Zhou, Y., Ling, S.: Aware attentive multi-view inference for vehicle re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018)
36. He, X. et al.: Triplet-center loss for multi-view 3d object retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018)
37. Lou, Y. et al. :Veri-wild: a large dataset and a new method for vehicle re-identification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019)
38. Bromley, J., et al.: Signature verification using a "siamese" time delay neural network. Adv. Neural Inf. Process. Syst. (1994)
39. Chopra, S., Raia, H., Yann, L.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 1. IEEE, (2005)
40. Taigman, Y. et al. Closing the gap to human-level performance in face verification. deepface. IEEE Comput. Vis. Pattern Recogn. (CVPR). **5**. (2014)
41. Paisios, N., Subramanian, L., Rubinsteyn, A.: Choosing which clothes to wear confidently: a tool for pattern matching. In: Workshop on Frontiers in Accessibility for Pervasive Computing. ACM. (2012)
42. Berlemont, S. et al.: Siamese neural network based similarity metric for inertial gesture classification and rejection. (2015)
43. Liu, X., et al.: Siamese convolutional neural networks for remote sensing scene classification. IEEE Geosci. Remote Sens. Lett. **16**(8), 1200–1204 (2019)
44. He, H., et al.: Matching of remote sensing images with complex background variations via Siamese convolutional neural network. Remote Sens. **10**(2), 355 (2018)
45. Liu, X., et al.: A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In: European conference on computer vision, Springer, Cham (2016)
46. Wen, L. et al.: UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. arXiv preprint arXiv:1511.04136 (2015)
47. He, K. et al.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. (2015)
48. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
49. Yang, L. et al.: A large-scale car dataset for fine-grained categorization and verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015)
50. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015)
51. Yuan, Y., Yang, K., Zhang, C.: Hard-aware deeply cascaded embedding. In: Proceedings of the IEEE international conference on computer vision. (2017)
52. Liao, S. et al.: Person re-identification by local maximal occurrence representation and metric learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015)
53. Xiao, T. et al.: Learning deep feature representations with domain guided dropout for person re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016)

54. Zhou, Y., Shao, L.: Cross-view GAN based vehicle generation for re-identification. BMVC. **1**,(2017)
55. Shen, Y. et al.: Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. In: Proceedings of the IEEE International Conference on Computer Vision. (2017)
56. Wu, Y. et al.: Detectron2
57. Chatfield, K. et al.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)

Kang Yin received the B.Eng. degree in Internet of Object Engineering from Hohai University, Nanjing, in 2020. He is currently pursuing the Ph.D. degree in the Department of Artificial Intelligence at Korea University. His research interests include computer vision and brain-computer interface.

Liantao Wang received the B.S. degree in mechanical engineering and the Ph.D. degree in pattern recognition and intelligent system from the Nanjing University of Science and Technology in 2004 and 2015, respectively. From 2012 to 2014, he was a Visiting Scholar with the Robotics Institute, Carnegie Mellon University. He is currently an associate professor with the College of Internet of Things Engineering, Hohai University. His research interests include machine learning and pattern recognition.

Jinxia Zhang received the B.S. degree from the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China, in 2009 and the Ph.D. degree with the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China, in 2015. She was a visiting scholar in Visual Attention Lab at Brigham and Women's Hospital and Harvard Medical School from 2012 to 2014. She is currently an associate professor in the School of Automation, Southeast University. Her research interests include saliency detection, knowledge transfer, computer vision and deep learning.