Knowledge-Based Systems 123 (2017) 102–115

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Passenger flow estimation based on convolutional neural network in public transportation system

Guojin Liu[a,*], Zhenzhi Yin[a], Yunjian Jia[a], Yulai Xie[b]

[a] College of Communication Engineering, Chongqing University, Chongqing 400040, China
[b] Hitachi (China) Research & Development Corporation, Beijing,100190,China

ARTICLE INFO

*Article history:*
Received 30 July 2016
Revised 9 February 2017
Accepted 11 February 2017
Available online 14 February 2017

*Keywords:*
Passenger flow estimation
Convolutional neural network
Biologically inspired pheromone map

ABSTRACT

Automatic passenger flow estimation is very useful in public transportation system, which can improve the efficiency of public transportation service by optimizing the route plan and traffic scheduling. However, this task usually encounters many challenges in public transportation system, such as low resolution, background clutter, variation of illumination, pose and scale, etc. In this paper we propose a passenger counting system based on the convolutional neural network (CNN) and the spatio-temporal context (STC) model, where the CNN model is used to detect the passengers and the STC model is used to track the moving head of each passenger, respectively. Different from the traditional hand-engineered representation methods, our method uses CNN to automatically learn the related features of passengers. Meanwhile, target pre-location is used by combining the mixture of Gaussian (MoG) model and background subtraction, which can greatly reduce the following detection time. To address the tracking drift problem, inspired by the movement of ants in nature, we attempt to exploit the trajectory information to build a biologically inspired pheromone map and a 3D peak confidence map. Then, the number of passengers can be obtained by counting the regions of interest (ROI). Experimental results on an actual public bus transportation dataset show that this method outperforms some existing methods.



## 1. Introduction

Automatic passenger flow estimation is very useful for traffic management and overcrowding situation detection in public transportation. The accurate passenger flow information can improve the efficiency of public transportation service by optimizing the route plan and traffic scheduling. Meanwhile, it can also prevent severe traffic accidents caused by overloading. Traditionally, automatic passenger counting can be done by contact-type counters, optical sensors, and vision-based systems. Contact-type counters can be applied in many public places with entrances, such as subways and bus stations. However, it can cause congestion when the passenger flow is high because it counts passengers in sequence one by one. Optical sensors, such as radiation beam systems, do not block the doorways, but suffer from the undercounting problem. In recent years, automatic method of counting the passing passengers based on digital image processing has attracted more attention, which can reduce the cost and require no user intervention.

In the past few decades, several methods related to people counting have been proposed. Generally speaking, these methods can be divided into three approaches. The first approach is trajectory clustering based counting. This kind of methods count passengers based on the following hypothesis. Those trajectories belonging to the same human body are more similar than trajectories belonging to different individuals. In [1], the trajectories of visual features were clustered, and the number of passengers was estimated by the number of these clusters. Based on Dirichlet process mixture models (DPMMs), Topkaya et al. [2] employed a clustering scheme to estimate the number of passengers. This method fused a set of spatial, color and temporal information features for each detection. However, the performance of these methods will degrade greatly in illumination variation and low resolution transportation scene since this situation usually reduces the stability of the algorithms. The second one is regression based counting. The number of passengers in this type of methods is estimated by learning the regression function between features extracted from the input images and the people counted in a scene. Many regression functions, such as Bayesian regression [3], neural networks [4,5] and SVR regression [6,7], were used to estimate the crowd density. However, these methods just provide the counted number of people and fail to locate the individuals. Sometimes, the location information of individuals is important for video surveillance security systems, es-

* Corresponding author.
*E-mail address:* liuguojin@cqu.edu.cn (G. Liu).

http://dx.doi.org/10.1016/j.knosys.2017.02.016
0950-7051/

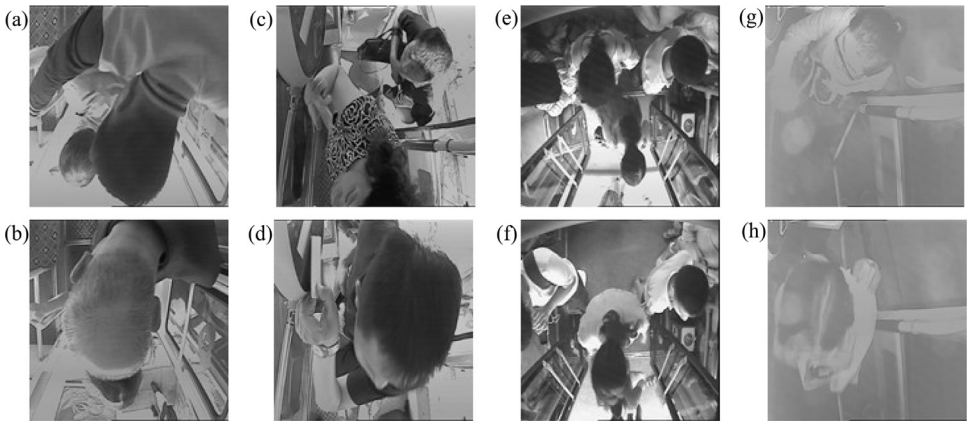


**Fig. 1.** Examples of the public transportation scenarios.

pecially in public transportation scenarios. The last one is detection based counting. In this type of approach, a detector is carefully designed to detect the people from the input images. Different type of detection methods, such as body detection [8–10], head-shoulder detection [11–13], skeleton graph [14,15] and head template matching [16,17], were proposed. Although this kind of methods offers the promising results, the carefully hand-engineered designed detector is not robustness in low resolution scene. It fails for some real surveillance applications. Besides, the stereo camera based method in [16] needs multiple cameras and extra calibration procedure, which might cause lots of inconvenience in deployment.

In recent years, although a few advances have been proposed [18,19] in this field, there are still some challenges, especially in complicated low-resolution scenes. The resolution of most videos in public transportation surveillance system is relatively low. Nonetheless, this task is nontrivial due to the illumination, occlusion, scale and pose variation of passengers in cluttered background. Some of them, such as the regression based method [19] are not suitable for our application. The number of passengers is time-varying in our scene. The examples of typical application are shown in Fig. 1.

From Fig. 1, we can observe that in addition to occlusion caused by the movement of passengers in Fig. 1(a) and (b), the images in Fig. 1(a)–(h) are of low resolution with 320*320 and the scales of heads in Fig. 1(c) and (d) change gradually as the passengers get on/off. Fig. 1(e) and (f) show the background clutter caused by the vibration of buses, which will be difficult for the followed-up counting. Meanwhile, lots of issues should be addressed due to the variation of illumination, as shown in Fig. 1(g) and (h). To our best knowledge, there are few papers discussing the problem of passenger counting in the complicated low-resolution public transportation scenarios.

Recently, convolutional neural networks (CNN) have shown outstanding performance on image classification tasks [20–22] and object detection tasks [23–26]. Different from traditional hand-engineered representations, they have deep architectures [27] and can learn powerful object representations. Motivated by this, we adopt CNN to handle the above mentioned challenges, which can extract the target feature automatically and is robust to the variations of illumination, pose and scale.

In this paper, we propose a passenger counting system to address the counting problem by combining the CNN detection model and the spatio-temporal context (STC) model [28], which offers the following advantages. Firstly, by combining the mixture of Gaussian (MoG) model [29] and background subtraction, target pre-location will greatly reduce the detection time of CNN. The CNN model is used to automatically learn the related features of passengers and then detect the moving passengers. Secondly, inspired by the movement of ants in nature, a biologically inspired pheromone map and a 3D peak confidence map are proposed to address the tracking drift.

The rest of the paper is organized as follows. Section 2 briefly describes the overview of our framework and introduces each module of the proposed method in detail. In Section 3, we evaluate our method with extensive experiments on an actual public transportation dataset. Finally, conclusions and future work are given in Section 4.

## 2. Passenger counting by CNN and STC model

### 2.1. Overview of our framework

As shown in Fig. 2, the proposed passenger counting framework includes two stages: the off-line training stage and the online counting stage. In the off-line training stage, the CNN model is trained by both the positive samples and negative samples with a new dataset constructed from the real complicated public transportation scenarios. The layers, weights and neuron numbers of CNN model are obtained by the off-line training. After that, the online counting stage is used to count the passengers in the videos, which includes five modules: the pre-location of passengers, passenger detection by CNN model, smallest enclosing circle clustering, STC tracking and biologically inspired pheromone map module. Each module is briefly shown in the following.

(1) Passengers pre-location: This phase extracts the moving pixels by MoG model and calculates the occupant's location from the binary map of moving objects. It can reduce the computational complexity of the object detection. Traditionally, off-the-shelf methods, such as sliding window based detection methods, are to provide the top $N$ candidate object regions, where $N$ is usually not smaller than 1000 [18,30,31]. Meanwhile, most of the candidate regions are not related to interested objects since the provided candidate regions are much more than real objects. Thus, this strategy will greatly increase the classification time of CNN. The comparison of the computational complexity between sliding window based detection and MoG based detection for our application is shown in Table 1. It is shown that although the run time of MoG is longer than that of sliding window, the following detection time of the former is much smaller than that of the latter. That is because the provided candidate regions of sliding window are much more than that of the MoG. Meanwhile, most of the scenes in the video are background. The detection phase of background with MoG will not be done in this situation, since there

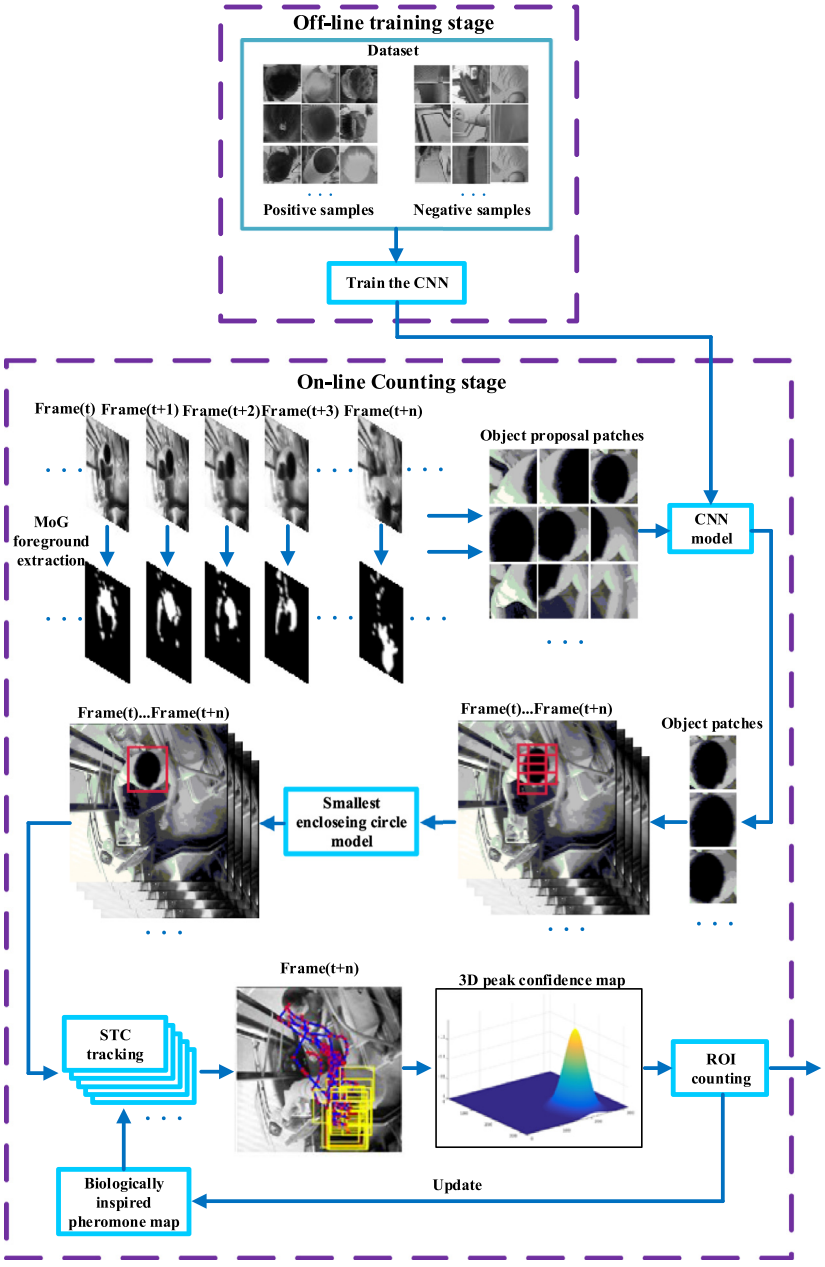


**Fig. 2.** The framework of the proposed passenger counting system.

**Table 1**
The comparison of the computational complexity between MoG based detection (Method 1) and sliding window based detection (Method 2).

| Modules | Runtime(seconds) per frame in method 1 | Runtime(seconds) per frame in method 2 |
|---|---|---|
| Pre_location | 0.618 s | 0.011 s |
| Detection | 0.996 s | 2.236 s |
| Total | 1.614 s | 2.247 s |

are no moving objects. In contrast, the detection phase of sliding window will be processed for all kinds of scenes.

(2) Passenger detection and smallest enclosing circle clustering model: After the pre-location step, the input frame is detected by six different CNNs trained in off-line, which detects the occupant from the background independently. It should be noted that the number of positive samples (34,322) is less than that of negative samples (179,398). Thus, the detection task can be considered as an imbalanced learning problem [32]. The fundamental issue with the imbalanced learning problem is the ability of imbalanced data to significantly compromise the performance of most standard learning algorithms. In order to solve this problem, we use random oversampling and undersampling [32] to divide the positive and negative samples into six patches. Each patch is used to train one CNN model. The six CNNs with different weights are trained independently.

**Table 2**
Parameters for our system.

| Parameters | Value |
|---|---|
| The number of samples for training | 213,720 |
| Positive samples | 34,322 |
| Negative samples | 179,398 |
| Size of sample | 32*32*1 |
| Layers of CNN | 7 |
| Learning rate | 0.15 |
| Batch size | 50 |
| Number of iteration | 20 |
| Batch normalization | Yes |
| Dropout ratio | 0.1 |
| The number of initial frames in MOG model | 100 |
| Size of candidates in MoG | 64*64 |

After detection by six different CNNs, more than one detection window will be presented for a specific passenger. The smallest enclosing circle clustering model is to aggregate the similar target detection windows into one target detection window, which can reduce the complexity of tracking and improve the robustness of tracking.

(3) Passenger tracking: Once the potential passengers are validated, the STC method is used to get the related trajectories, which formulates the spatio-temporal relationships between the object of interest and its local context with a Bayesian framework. Meanwhile, STC models the statistical correlation between the low-level features (i.e., image intensity and position) and its surrounding regions.
(4) Biologically inspired pheromone map and ROI counting module: Based on the trajectories formed previously, a biologically inspired pheromone map is proposed to address the tracking drift problem. Then, based on the overlapped tracking window, a 3D peak confidence map is built. The entry/exit line is set based on a rectangular area. The algorithm detects the peak of counting wave. Once the peak is greater than the threshold, the counting switch is triggered and the movement of wave peak can be detected. When the wave peak is approaching to the edge of entry/exit line, the counted number increases by one.

### 2.2. Passenger pre-location

The pre-location of passengers is used to reduce the computational complexity of object detection by giving the related position of target. Background subtraction technique is traditionally applied to the pre-location of objects. However, it detects not only objects but also a lot of noise in a low resolution surveillance system since it shows great sensitivity to small changes, such as variations of illumination and background. Meanwhile, the variation of occupant's pose and scale will also influence the background subtraction.

Here, MoG is used to approximate the background due to its capability of modeling multiple model backgrounds. The MoG method can be described as follows. The values of a particular pixel{$x$, $y$}over time $t$ are considered as a "pixel process", which is a time series of pixel values, e.g. scalars for gray values or vectors for color images. The recent history of each pixel{$X_1$, ..., $X_t$}can be represented by a mixture of Gaussian distributions,

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * y(X_t, u_{i,t}, \sum i, t) \quad (1)$$

where $K$ is the number of distributions, $w_{i,t}$, $u_{i,t}$ and $\Sigma i,t$ are the weight estimation, mean value and covariance matrix of the $i^{th}$ Gaussian in the mixture at time $t$, respectively, and $y$ is a Gaussian probability density function with mean of $u_{i,t}$ and variance of $\Sigma i,t$.

Based on MoG model, we get the binary moving region map of each frame. Then, a simple morphological operation is applied to fill holes and smooth blobs. The major process of morphology processing is as follows.

- Step 1: fill holes in the binary image. A hole is a set of background pixels that can not be reached by filling in the background from the edge of the image.
- Step 2: create a flat disk-shaped structuring element with a radius of 3 and erode binary image with the above flat disk-shaped structuring element.
- Step 3: create a matrix template $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ and dilate binary image with the matrix.

Finally, we get the binary map of moving target for each frame. Fig. 3 shows the corresponding results of moving target detected by the pre-location phase. It can be seen that the proposed method can effectively locate the moving target. Fig. 3(b) is the background extracted by MoG model, which shows that the background is consistent with the real scene. Fig. 3(c) is the foreground with the approximated shape of the passengers. Fig. 3(e) shows the corresponding binary pre-location result after morphological operation. It can be seen that the position of target can be obtained by pre-location. Different from the sliding window, which segments each frame into many candidates and is usually time-consuming, our method can dramatically reduce the amount of passenger candidates and improve the detection efficiency.

### 2.3. Passenger detection by CNN

The popular CNN model is Alex model on a large scale dataset, such as ImageNet [33]. Unfortunately, the common CNN architecture is not suitable for our case due to the following reason. It usually has a very deep structure, which will cause over-fitting concerns for our limited surveillance dataset. Before the target detection by CNN, the first issue is to design an architecture with suitable layers, weights and neurons of the network for our specific application.

Based on extensive experiment results, the architecture in our experiments is shown in Fig. 4. The size of the input layer is 32*32 pixels. Our CNN architecture generally has three convolution layers, five batch normalization layers, three max-pooling layers and two fully-connected layers. The convolution layer 1 has 35 filters with the size of 5*5, the convolution layer 2 has 70 counterparts with the size of 3*3, while the convolution layer 3 has 105 counterparts with the size of 3*3. These three layers are configured to act as a hierarchical feature extractor, which captures low-level features. Feature maps are usually first followed by a batch normalization layer [34] and then followed by max-pooling layer with 2*2 kernel. This improves generalization, robustness to small distortions and also reduces the dimensionality. A higher level feature map takes its input from several lower level maps. In such a way, a hierarchical feature model can be implemented. For each convolution and pool layer, a rectified linear unit (ReLU) is also adopted to further generate high-level nonlinear feature responses. Next, two additional fully connected layers with a softmax classifier are used to output the detection result for a specific input image.

The standard stochastic gradient descent (SGD) algorithm [35] with the cross-entropy error function is used to learn the parameters of the proposed CNN model. Based on the extensive experiments on recall and accuracy, a fixed learning rate with the value of 0.15 is used for our application. In order to improve the generalization ability of the network and avoid the over-fitting issue, the dropout regularization [36,37] and batch normalization techniques are combined to get an effective training model with

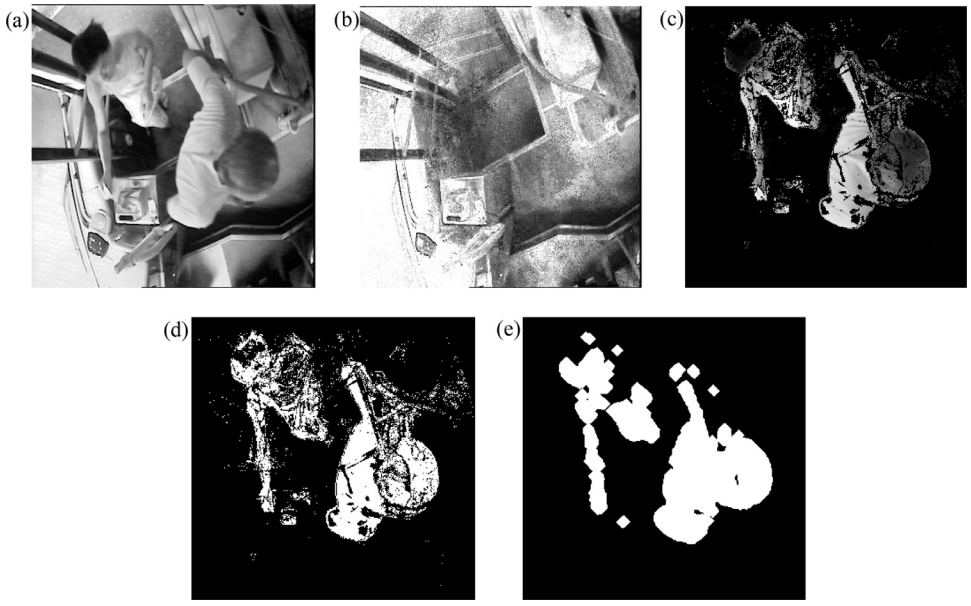


**Fig. 3.** The original image and its pre-location results, (a) original frame, (b) background extracted by MoG model, (c) foreground, (d) binary map of foreground, (e) pre-location map after morphological operation.

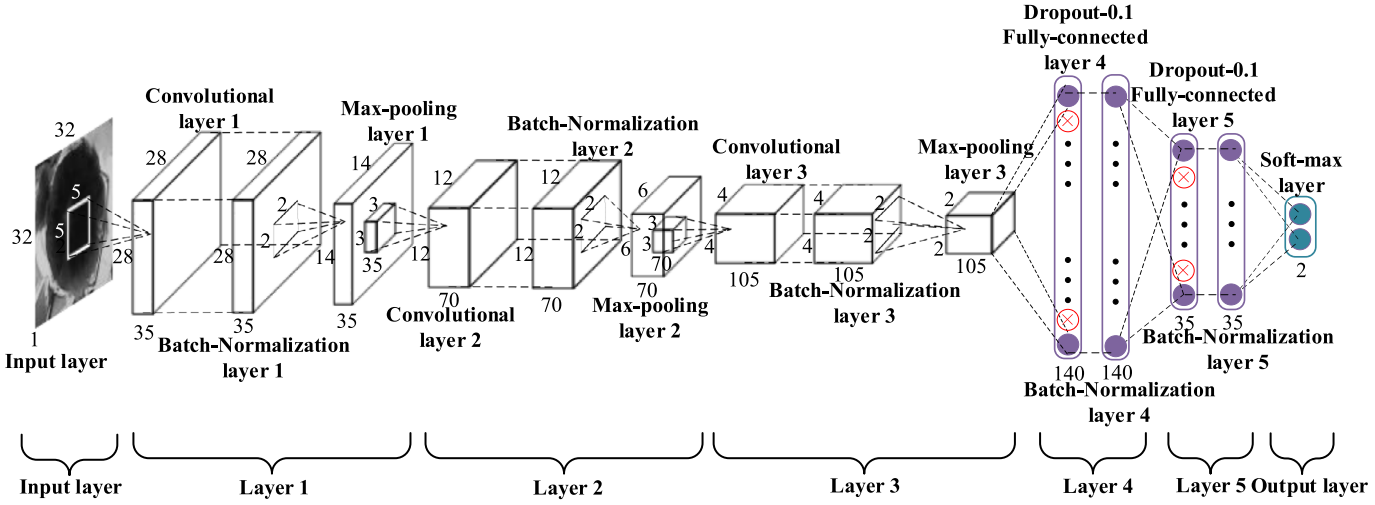


**Fig. 4.** The CNN architecture of the proposed method.

better performance. In the off-line training stage, the samples are divided into six independent patches to train independent CNN models, where the positive samples are selected with replacement and the negative samples are selected without replacement. When the moving target has been detected by all the CNN models, it can be considered as a target candidate. Then, the target candidate is used as an input for the following tracking step. Otherwise, the target candidate is viewed as background. Although it may be a little more complexity than one CNN model, the proposed method is robustness to target detection in the condition of imbalanced data.

After detection by six CNN models, more than one detection window is obtained. In order to identify moving passengers, the smallest enclosing circle is used to aggregate the above detection windows into one detection window. The aggregate algorithm can be described as follows. Given the coordinates of $n$ points, the smallest enclosing circle model is to find a circle with smallest radius, where all the $n$ points are surrounded by the circle. In our application, the points in the plane are the center points of the detection rectangular windows. After getting the circle with smallest radius, the circumscribed rectangle of the above circle is considered as the detection window of moving target.

The aggregation result of detection windows is shown in Fig. 5. From Fig. 5, it can be seen that although more than one detection windows are shown in the figure, the final detection result of CNN model is only one. It will be useful for the following tracking stage.

### 2.4. Passenger tracking

After passenger detection by CNN, the STC model is used to track the moving target, which can solve the variation problem of head scale in our scenes. It adapts itself to the spatio temporal contexts by modeling the relationship of its local context regions between time and space. The tracking algorithm is divided into three steps:

- A spatial context model between the target object and its local surrounding background is learned based on their spatial correlations in a scene.

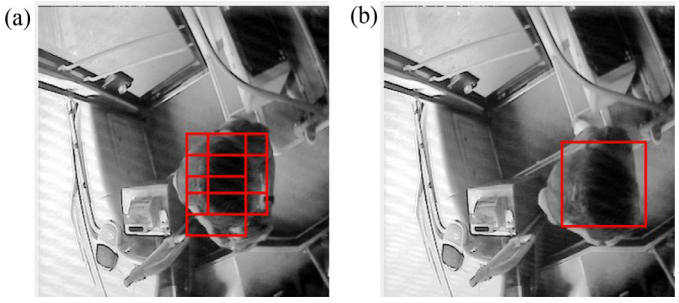


**Fig. 5.** Detection window before and after aggregation by smallest enclosing circle, (a) before smallest enclosing circle clustering, (b) after smallest enclosing circle clustering.

- The learned spatial context model is used to update a spatio-temporal context model for the next frame. The tracking problem is formulated by computing a confidence map which estimates the likelihood of object location.
- Based on the estimated confidence map, STC model selects the position with maximum likelihood probability as the best target location and begins to track.

For our application, the passenger location in the first frame has been initialized by our CNN model and the smallest enclosing circle algorithm. At the $t$-th frame, we learn the spatial context model $h_t^{sc}(x)$, which is used to update the spatio-temporal context model $H_{t+1}^{stc}(x)$to reduce noise introduced by target appearance variations. $H_{t+1}^{stc}$ is then applied to detect the object location in the $(t+1)$-th frame. When the $(t+1)$-th frame arrives, the object location$x_{t+1}^*$in the $(t+1)$-th frame is determined by maximizing the new confidence map

$$x_{t+1}^* = \arg\max m_{t+1}(x) \tag{2}$$

where $m_{t+1}(x)$ is represented as

$$m_{t+1}(x) = H_{t+1}^{stc}(x) \odot I_{t+1}(x)\omega_{\sigma t}(x - x_t^*) \tag{3}$$

Here, $I(\cdot)$ is image intensity that represents appearance of context, $\omega_\sigma(\cdot)$is a Gaussian weighted function and $\odot$represents the convolution operator. Fig. 6 shows the STC tracking architecture [28], where $\rho$ is a coefficient between 0 and 1, $h_t^{sc}$is a spatial context model in frame $t$, and$h_t^{stc}$is a spatio-temporal context model in frame $t$. More details about STC model can be found in [28].

### 2.5. Biologically inspired pheromone map and ROI counting

Although STC model is good in most cases, it is sensitive to the initial position of the object. Meanwhile, due to mutual interference between the surrounding background and different appearance of passengers, tracking drift in the successive frame grows over time. The tracking drift and correction are shown in Fig. 7. The first row is the illustration of the original passenger tracking, where the passenger is boxed by red dashed line. The second row is the illustration of tracking without tracking drift correction. From it, we can see that the tracking window in Fig. 7(e) is intensive and overlapped. In contrast, the tracking window is divergent in Fig. 7(f) and (g), where the yellow tracking window drifts from the moving passengers. In Fig. 7(h), there is only one yellow tracking window, which will cause undercounting problem. The third row is the illustration of tracking with tracking drift correction. From it, we can see that the tracking in Fig. 7(i) is as effective as the tracking in Fig. 7(e). In Fig. 7(j) and (k), once the tracking drift is detected, a biologically inspired pheromone map is used to correct it, where the correct tracking window is represented by green window. In Fig. 7(l), it can be observed that after tracking drift correction, our method can relocate the moving passengers and track it effectively.

Based on the above discussion, a biologically inspired pheromone map is proposed to overcome tracking drift. When the ants look for food or move outside, they will leave a lot of pheromone in the way they are moving. As for our scene, when the passengers get on/off, they will have a similar trajectory. Inspired by this, a biologically inspired pheromone map is designed. The more passengers passed by the entrance/exit area, the more pheromone will be left in the related regions. The process is shown in Fig. 8. Original frame with tracking trajectory is represented by blue line (Fig. 8a). The red star marker in the blue trajectory line represents the center point of each tracking window. From it, we can see that few passengers get on/off in the beginning, such that the pheromone is sparse and the shape of color map was rectangle. Gradually, with the increase of passengers getting on/off, the trajectory of pheromone map will be clearer than before, as shown in Fig. 8(b)–(f). Eventually, the pheromone map in Fig. 8(f) is built.

In the end, we correct the tracking drift with the biologically inspired pheromone map. The center position with the largest pheromone information is searched in the vicinity of the region. Based on the center position, the tracking window far away from the position is considered as a tracking drift window. We will delete the tracking window and move its center to the found center. Then we update the correct tracking drift window. The illustration of tracking drift correction is shown in Fig. 7(j) and (k),where the red window represents detection window, the yellow window represents tracking window and the green window represents correction tracking window. It can be observed that tracking drift exists in Fig. 7(f) and (g). After correction,

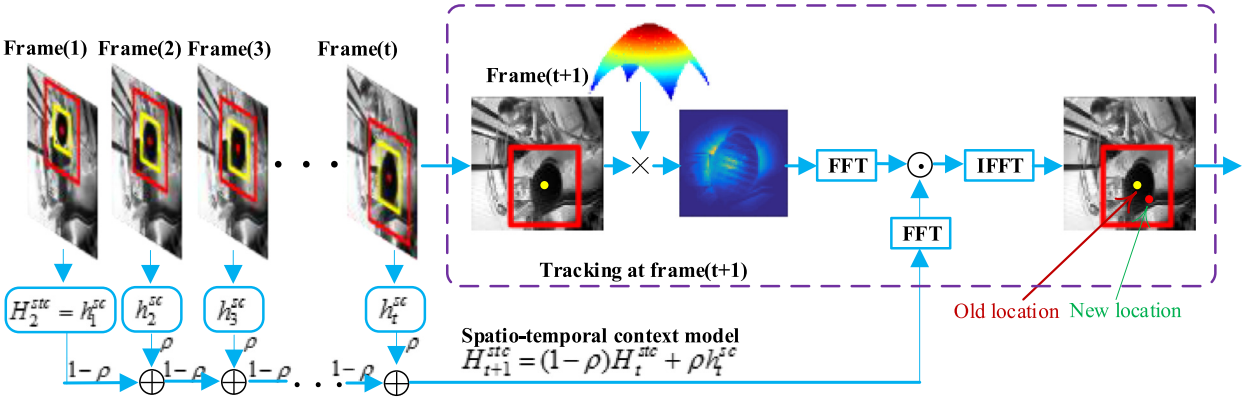


**Fig. 6.** STC tracking architecture.

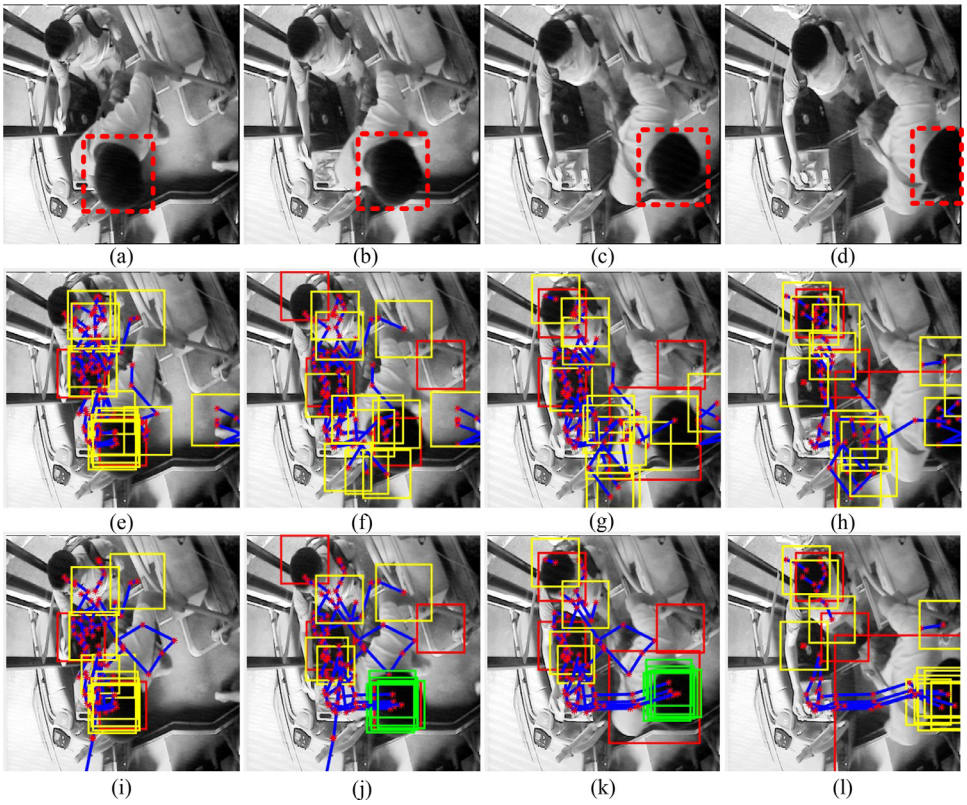


**Fig. 7.** Tracking drift illustration of passenger tracking, (a)-(d) are the original consecutive frames, (e)-(h) are the tracking trajectory of consecutive frames without correction, respectively. In contrast, (i)-(l) are the corresponding tracking trajectory of consecutive frames with correction, respectively. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

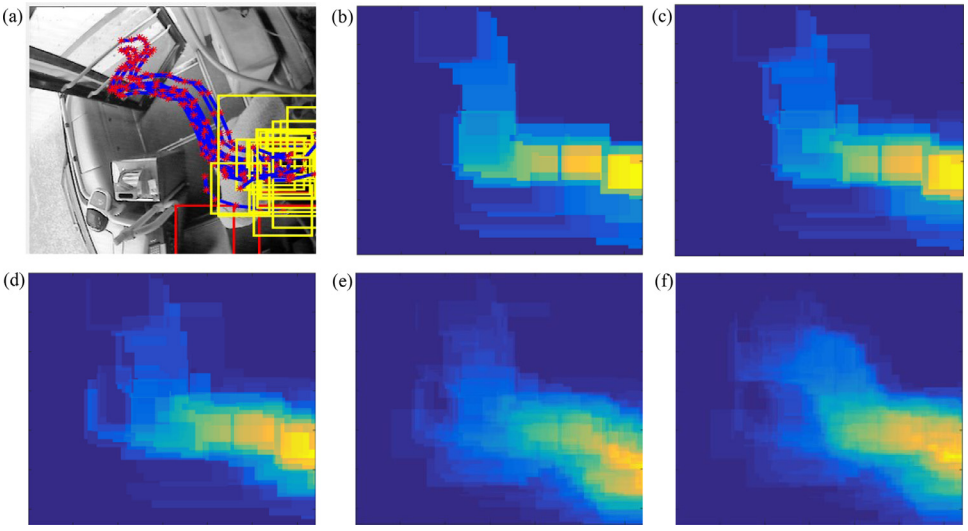


**Fig. 8.** Illustration of biologically inspired pheromone map generation, (a) original frame image with tracking trajectory represented by blue line, (b)-(f) are the corresponding biologically inspired pheromone maps in the followed up tracking process. (For interpretation of the references to color in the legend, the reader is referred to the web version of this article.)

the passenger can be effectively tracked, as shown in Fig. 7(j) and(k). The green box (Fig. 7(j) and (k)) shows the position of the passenger.

After tracking drift correction, a Gaussian-like distribution with the maximum value in the center point of the tracking window is built for each tracking window. The overlapped positions of multiple tracking windows are formed into a 3D peak confidence map, as shown in Fig. 9. The yellow bounding box shows the location of the global peak in the confidence map. For each pixel in $t$th frame, the confidence map can be represented by

$$C(i,j) = \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi}\sigma^{(k)}} \exp\left[-\frac{1}{2}\left(\frac{(i-u_i^{(k)})^2 + (j-u_j^{(k)})^2}{(\sigma^{(k)})^2}\right)\right] \quad (4)$$

where $C(i, j)$ is the value in the coordinate$(i,j)$, $K$ is the number of tracking windows, and $u_i^{(k)}$and $u_j^{(k)}$are the row and column of

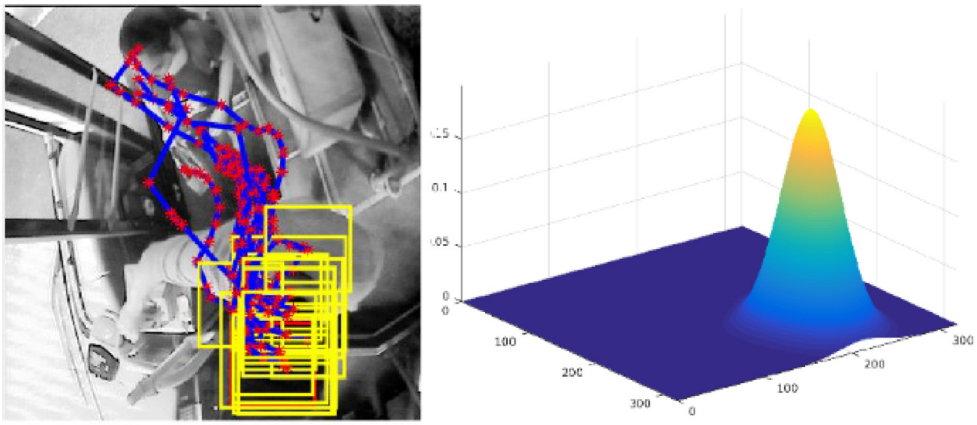

**Fig. 9.** Tracking diagram and its corresponding 3D peak confidence map. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

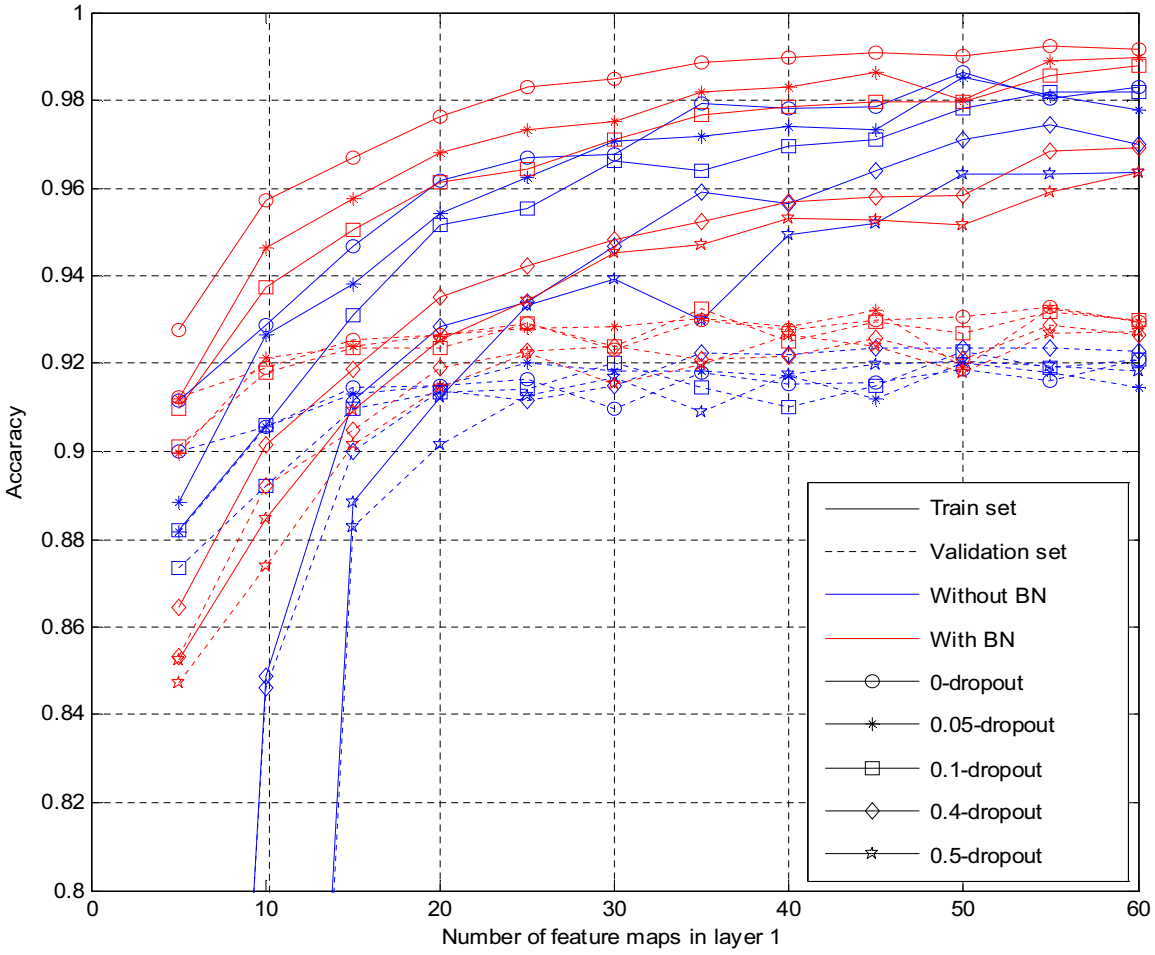


**Fig. 10.** Results of different parameters (dropout, batch normalization, number of convolution layer 1) on train set and validation set. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

center point in the $k$-th tracking window, respectively. In addition,$\sigma^{(k)} = r^{(k)}/6$, where $r^{(k)}$ is the length of tracking window. The targets in different spatial positions can be distinguished by the spatial and temporal difference in the peak confidence map. It can be observed that when there are a lot of overlapped tracking windows in the spatial position, the peak in the peak confidence map is high.

After the generation of confidence map, a rectangular window with the width of head is used as the counting area of people in/out, as shown in Fig. 15. When a peak of confidence map is detected in the window and the peak value is greater than the threshold (average peak value of 4 tracking windows), the counting switch is triggered to detect the movement of peak. When the crest is out of the counting edge region, the counted number

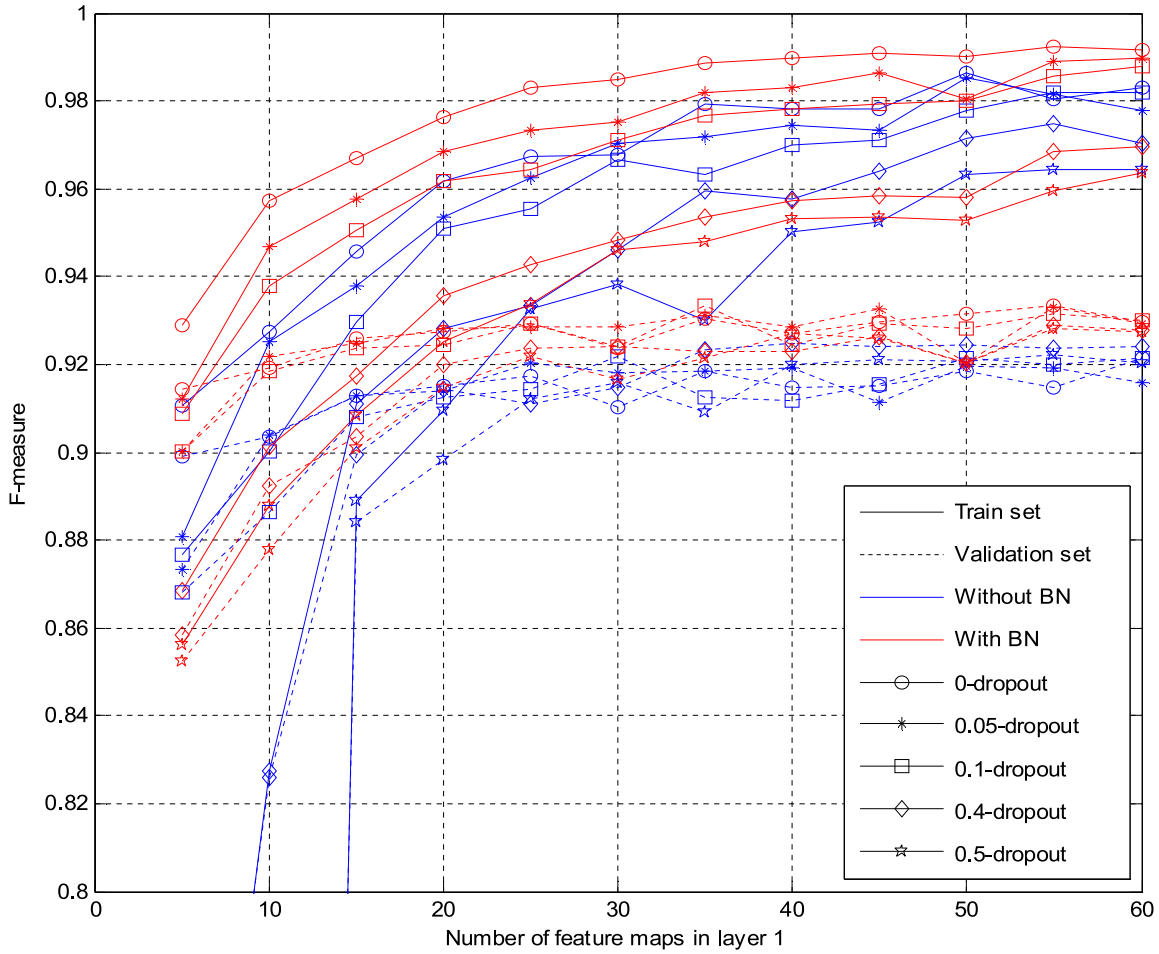


**Fig. 11.** F-measure results of different parameters (dropout, batch normalization, number of convolution layer 1) on trainset and validation set.

increases by one. After that, the switch is closed to remove the vibration of repeating count.

## 3. Experimental results

Our dataset is collected from surveillance video of public bus transportation in China. The classical scenes of our application are shown in Fig. 1. We manually extract 34,322 head positive samples and collect 179,398 negative samples from real surveillance video in bus transportation. We will show our counting demo video with front door, rear door and bi-directional counting scenes in the supplemental materials.

### *3.1. CNN architecture designation*

As a popular method, CNN has been widely used in image processing. However, different applications have different architectures. It is difficult to design the number of layers, the weights of each neurons and the number of neurons with common parameters. For our application, we evaluate the CNN architecture by extensive experiments. 50,000 training samples and 10,000 validated set samples with equal number of positive and negative samples are selected to train the CNN architecture. Fig. 10 shows the relationship between the accuracy and the number of feature maps in convolution layer 1. For the sake of simplicity, we just select the number of feature maps in the second and the third convolution layers as the double and triple of the first layer, as shown in Fig. 4, respectively. If the number of feature maps in first convolution layer changes, the other convolution layers will be set based on the above law. In Fig. 10, each curve is represented by three parameters: line color, line style (solid/dashed) and marker style. For example, the curve with red color, solid line and star marker is represented as a curve with batch normalization and a dropout ratio of 0.05 in the train set. In contrast, the curve with blue color, dashed line and square marker is represented as a curve without batch normalization and a dropout ratio of 0.1 in the validation set.

The overall algorithm of our method is shown in Algorithm 1.

From Fig. 10, we can see that with the increase of feature maps in convolution layer 1, the accuracy of trainset is close to 99%, while for validates set the accuracy is up to 93%. However, training process with different parameters, such as batch normalization and dropout, will have different impacts on the final result. The training process in trainset with batch normalization achieves better results than that without batch normalization. Meanwhile, the results on validation set also show similar performance. Based on the experiments, five batch normalization layers were used in our CNN architecture, as shown in Fig. 4.

When considering about the impact of dropout on the trainset and validate set, Fig. 10 shows the impact of different dropout ratio on accuracy in our method. It can be observed that although 0-dropout ratio is better than 0.1-dropout ratio in trainset in most cases, it is poorer than 0.1-dropout ratio in the validate set. Meanwhile, we can observe that the accuracy of our method with 0.1-dropout ratio is better than those with other dropout ratios. According to this experiment, we choose 0.1-dropout ratio in the full connected layer.

As for the selection of the number of feature maps in our CNN model, the easy way is to select more feature maps in each

**Algorithm 1**
Overview of our passengers counting method.

**Input**: Each frame of video image;
**Output**: Detection and tracking results for each frame of video image; Results for counting;
**Algorithm begin:**
MoG foreground extraction;
**for** k = 1...*K*th frame images **do**
  Update Gauss mixture model;
  Object proposals;
  Object detection;
  Smallest encloseing circle for target clustering;
  **if** the head target is detected **do**
    **for** *n* = 1...N targets **do**
      Add STC tracking model;
    **end for**
  **end if**
  STC model group for tracking one frame;
  Monitor tracking drift;
  **if** tracking drift exists **do**
    Update STC model group by biologically inspired pheromone map;
  **end if**
  Establish 3D peak confidence map;
  **if** peak is valid **do**
    ROI counting head target based on confidence map;
    Update biologically inspired pheromone map;
  **end if**
**end for**

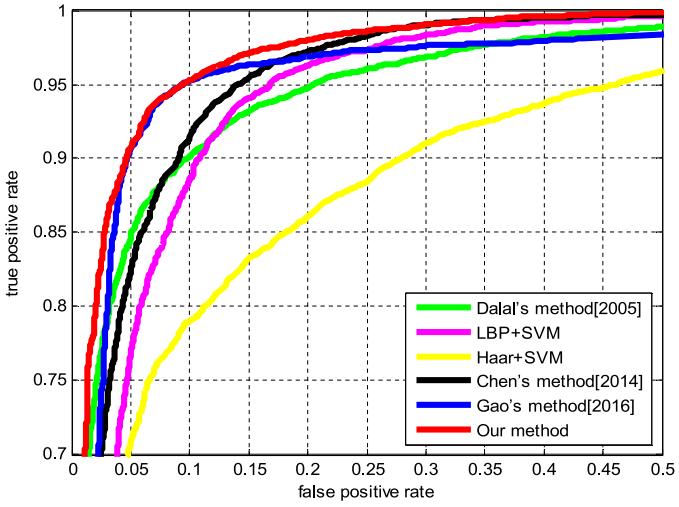


**Fig. 12.** ROC curves of six different methods.

layer, but it will cause overfitting and the architecture will be too complexity to train. Meanwhile, more feature maps will not result in better accuracy of validation set, as shown in Fig. 10. According to this experiment, we choose a stable and simple network with 35 feature maps in layer 1, as shown in Fig. 4.

The above results of our CNN architecture can be verified by the relationship between the *F*-measure and the number of feature maps in layer 1, as shown in Fig. 11. The *F*-measure is defined as the following equation

$$F-\text{measure} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

where the *F*-measure is a measure of the accuracy of validation set. It can be interpreted as a weighted average of the precision and recall. *F* score reaches its best value at 1 and the worst at 0.

### 3.2. Passenger detection

After the designation of CNN architecture, we adopt it to detect the moving passengers. Table 3 shows several detection results on our public transportation dataset. We use three indices, namely accuracy, recall rate and precision on validation set, to evaluate the following six methods, that is, Dalal's method [10], LBP+SVM, Haar+SVM, Chen's method [13], Gao's method [18] and our method. Precision, recall and accuracy are defined as the following equations, respectively.

$$\text{precision} = \frac{t_p}{t_p + f_p}, \tag{6}$$

$$\text{recall} = \frac{t_p}{t_p + f_n}, \tag{7}$$

$$\text{accuracy} = \frac{t_p + t_n}{t_p + f_p + t_n + f_n}, \tag{8}$$

where $t_p$means true positive, $t_n$means true negative, $f_n$means false negative and $f_p$means flase positive.

From Table 3, it can be observed that although Haar+SVM and Chen's method have more feature dimensions than the other four methods (Dalal's method , LBP+SVM, Gao's method and our method), its accuracy, recall and precision are smaller than those of the other four methods, respectively. Compared with the five methods, it can be seen that although our method has less feature dimension than the other four methods (Dalal's method, LBP+SVM, Haar+SVM and Chen's method), it achieves better result than the other four methods. The accuracy of our method is up to 93.24 and the recall rate is up to 94.215. In contrast, the best result of the other four methods is only 90.8. The result is also verified by the precision of different methods. When compared with Gao's method, it can be seen that our method is better in terms of accuracy and recall indices. Although the precision index of Gao's method is higher than that of our method, the *F*-measure of our method is a little larger than that of Gao's method. That is to say, our method has better detection performance than Gao's method.

Fig. 12 shows the ROC curves of the above six methods. It can be observed that our method is better than the other five methods in passenger detection. In order to show the detection results intuitively, different detection maps detected by the above six methods are depicted in Fig. 13. It can be seen from the figure that our method has the better results than the other five methods.

**Table 3**
Results of different detection methods on our public transportation dataset.

| Detection methods | Accuracy | Recall | Precision | F-measure | Features |
|---|---|---|---|---|---|
| Dalal's method (HoG+SVM) | 90.38 | 88.5288 | 92.0422 | 90.2513 | 324 |
| LBP+SVM | 89.71 | 90.8549 | 88.9278 | 89.8810 | 256 |
| Haar+SVM | 84.23 | 81.3121 | 86.5242 | 83.8372 | 1024 |
| Chen's method (multi-feature+SVM) | 90.8 | 91.7893 | 90.1054 | 90.9396 | 1604 |
| Gao's method | 93.09 | 92.743 | **93.468** | 93.1041 | 35 |
| Our method | **93.24** | **94.215** | 92.486 | **93.3425** | **35** |

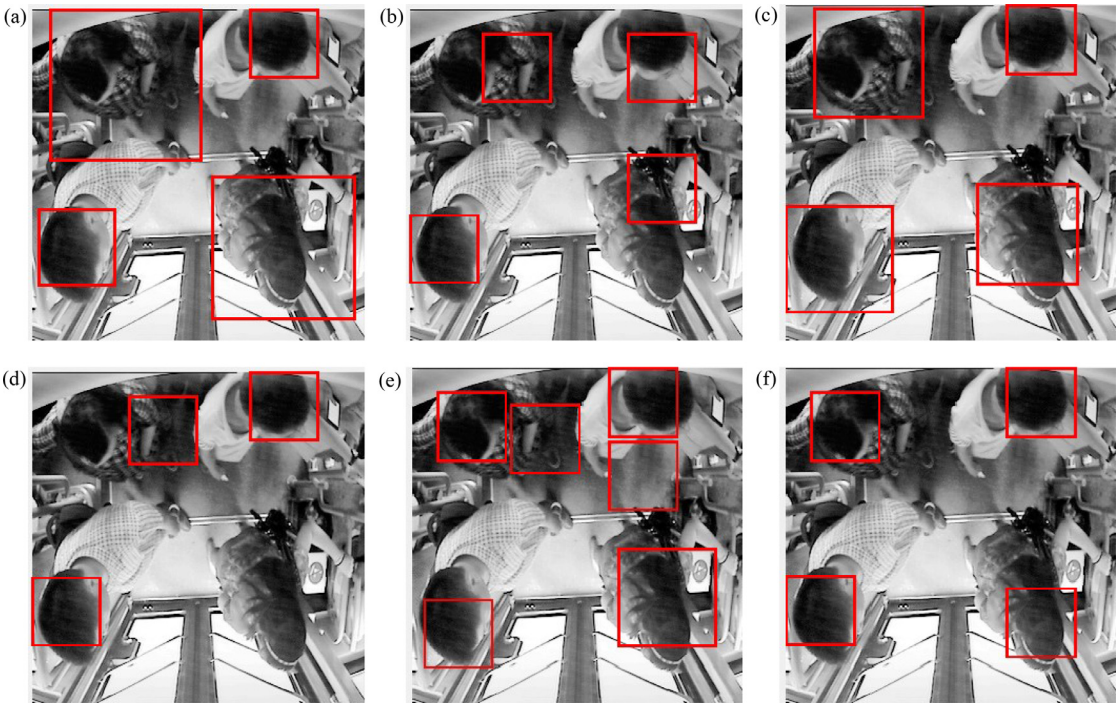


**Fig. 13.** Detection results of six different methods (Dalal's method, LBP +SVM, Haar+SVM, Chen's method, Gao's method and our method), (a) Detection by Dalal's method, (b) Detection by LBP+SVM, (c)Detection by Haar +SVM, (d) Detection by Chen's method, (e) Detection by Gao's method, (f) Detection by our method.

**Table 4**
Results of different counting methods on real public transportation dataset.

| Methods | With tracking drift correction | | Without tracking drift correction | |
|---|---|---|---|---|
| | The number of the count | ratio | The number of the count | ratio |
| Dalal's method | 98/116 | 84.48% | 66/116 | 56.90% |
| LBP+SVM | 94/116 | 81.03% | 65/116 | 56.03% |
| Haar+SVM | 80/116 | 68.96% | 63/116 | 54.31% |
| Chen's method | 101/116 | 87.07% | 75/116 | 64.66% |
| Gao's method | 103/116 | 88.79% | 74/116 | 63.79% |
| Our method | **108**/116 | **93.10**% | **75**/116 | **64.66**% |

### 3.3. Results on passengers counting

Based on the prior knowledge of our method, we applied the proposed method to determine the passenger flow in bus transportation scene. The relationship between the number of passengers and the image frame is shown in Fig. 14. Ground-truth was built by manually counting the passengers getting on/off the bus in surveillance video.

To quantitatively evaluate the performance of the proposed method compared with other methods, we applied six algorithms (Dalal's method, Chen's method, the present method, Gao's method, LBP+SVM and Haar+SVM) to 12 video clips in real bus transportation scenes, as shown in Fig. 14 and Table 4. In order to ensure the diversity, the 12 video clips were taken from different scenes including day and night, crowded and uncrowded, etc. All the algorithms are tested by two situations: with and without tracking drift correction. There are 1636 frames in the above experiment and the ground truth of passengers is 116. 108 passengers have been counted in our method with tracking drift correction and the counting rate is up to 93.1%. While for Gao's method, 103 passengers are counted with the counting rate of 88.79%. For the other four methods, the counted passengers are 98 (Dalal's method), 101 (Chen's method), 94 (LBP+SVM) and 80 (Haar+SVM), respectively. Based on the above analysis, it is obvious that the effectiveness of our method is better than the other five methods.

When considering the situation without tracking drift correction, 75 passengers have been counted in our method with the counting rate of 64.6%. The best counting result of the other five methods is 75, which is achieved by Chen's method. From it we can see that tracking drift correction by biologically inspired

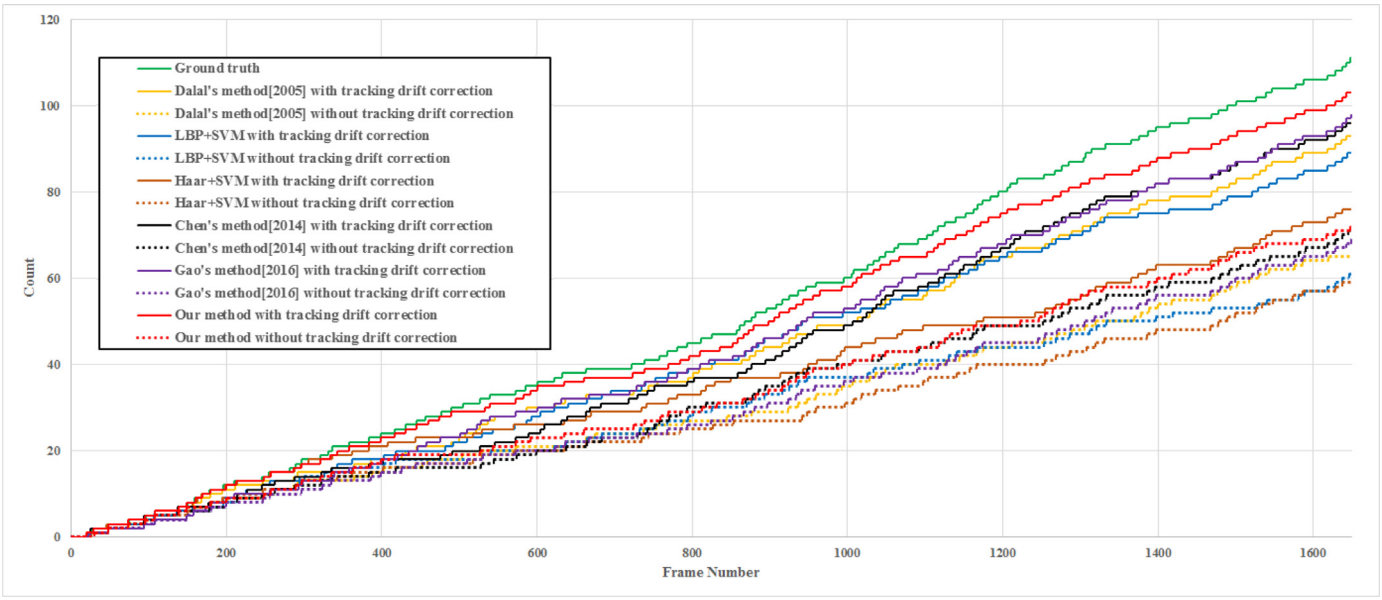


**Fig. 14.** The comparison of our passengers counting method and the other five methods on the test set with 1636 test images.

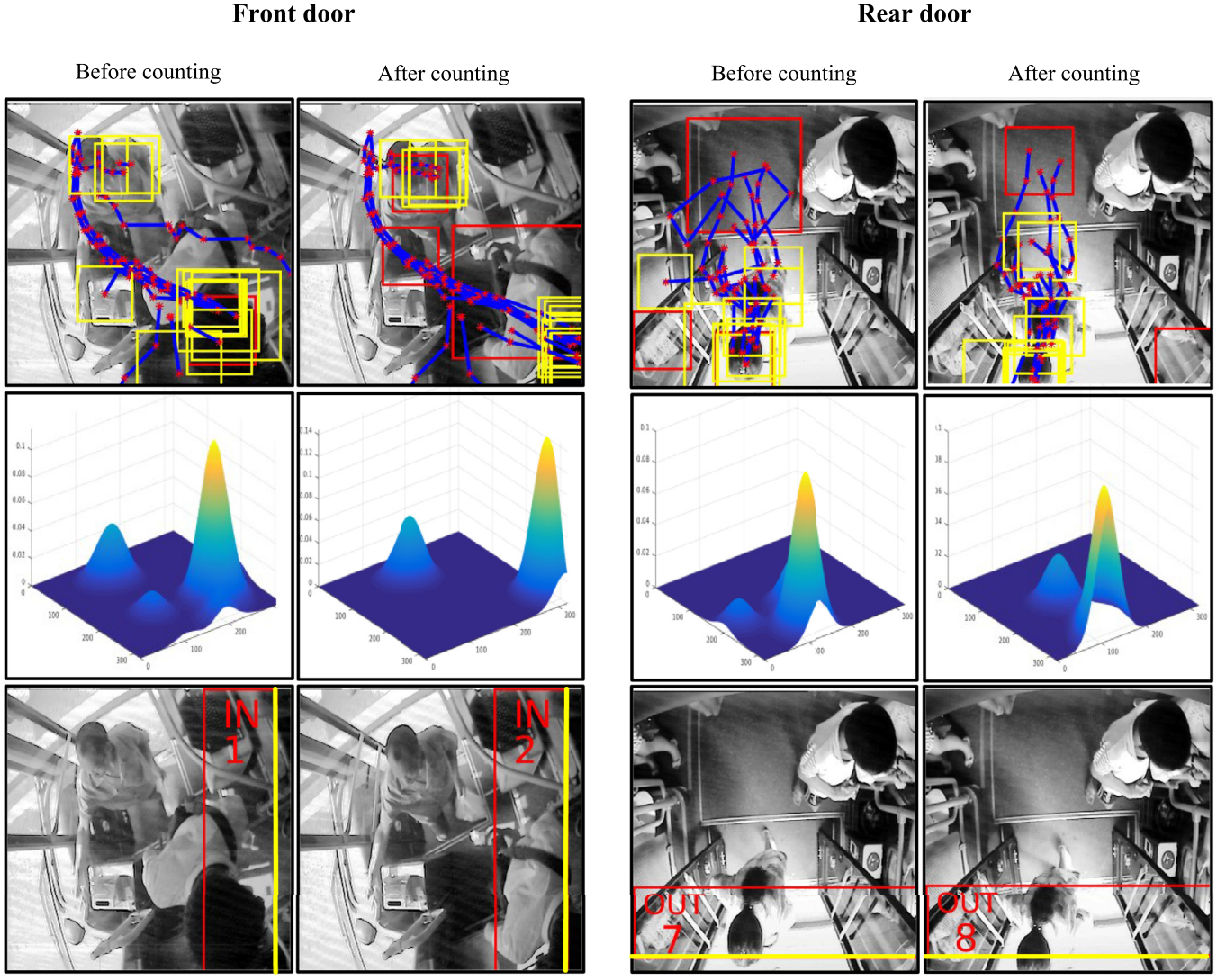


**Fig. 15.** The illustration of passenger counting in real scenes. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

**Table 5**
The computational complexity of our method.

| Modules | Runtime(seconds) per frame in Matlab |
|---|---|
| Pre_location | 0.618 |
| Detection | 0.996 |
| Tracking+Counting | 0.178 |
| Whole system | 1.792 |

pheromone map is important for the track. Meanwhile, the capacity of feature representation with CNN methods (the proposed method and Gao's method) are better than those of the other four methods. The reason is that CNN can learn more complex models than shallow ones.

Fig. 15 shows the passenger counting process before counting and after counting in front door and rear door. The first row in the figure shows the red detection window and the yellow tracking window, respectively. The second row is the Guassian-like distribution based on the accumulation of yellow tracking window in first row, respectively. The third row shows the corresponding counting result, where the red rectangle area represents the counting area. The yellow line in the third row is the entry/exit line. The counting number is added based on the variation of wave peak, as shown in the second row. It can be observed that our method can effectively count the passengers in public transportation scenarios. More results about the passenger counting demonstration, including bi-directional counting in front door and so on, can be found in the supplemental materials.

Table 5 shows the computational complexity of our method in Matlab without any code optimization. The complexity is calculated by averaging the runtime of the above1636 frames, where passengers are present in nearly 90% frame. Since six CNNs are independent and can be executed simultaneously, we only calculate the runtime of one CNN detection model. Our method is tested in a desktop with an Intel i3 3.5 GHz and 16 GB RAM. It is shown in [38] that Matlab code is about 10 times slower than C++ code in terms of execution time. For the comparison between CPU and GPU, [39] shows that GPU is about 3∼20 times faster than CPU in terms of execution time for different benchmarks. Furthermore, paper [40] has shown that deep learning can be performed in real time by exploiting both the CPU and DSP of a mobile device SoC. Based on the above analysis, after optimization by C/C++ programming language and embedded system on the public transportation bus/subways, our method can meet the computational demand in real public transportation scenarios.

## 4. Conclusion

The proposed system uses a single inexpensive camera mounted overhead, which eliminates the need for calibration and creates a low-cost system. The presented passenger counting system combines the CNN detection model and spatio temporal context model to address the counting problem in the scenes of low resolution and with variation of illumination, pose and scale. Different from many sliding windows based detection method, our method adopts MoG model to obtain the foreground, which can dramatically reduce the detection time. After that, CNN model is trained to detect the potential moving passengers. To ensure the robustness of our method, we propose a biologically inspired pheromone map to correct the tracking drift. Finally, a 3D peak confidence map is used to count the passing passengers. Experimental results show that this method has better performance than many existing methods. In the future, we will extend current method with more deep learning algorithm, such as highway deep network [41–43] and extreme learning machine [44,45], to obtain better result with faster speed. It may be pointed out that because of the usage of detection and tracking algorithm in our application, it may not work in a very dense crowd scene. For this case, crowd density map estimation with deep learning will be explored in our future work.

## Acknowledgement

Thank Hitachi (China) Research & Development Corporation for providing the foundation for this research. This work was supported partly by the National Natural Science Foundation of China with Grant No. 61671452, No. 61675036 and No. 61302054.

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.knosys.2017.02.016.

## References

[1] G. Antonini, J.P. Thiran, Counting pedestrians in video sequences using trajectory clustering, IEEE Trans. Circuits Syst. Video Technol. 16 (8) (2006) 1008–1020.
[2] I.S. Topkaya, H. Erdogan, F. Porikli, Counting people by clustering person detector outputs, in: Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on, IEEE, 2014, pp. 313–318.
[3] A.B. Chan, N. Vasconcelos, Counting people with low-level features and Bayesian regression, Image Process. IEEE Trans. 21 (4) (2012) 2160–2177.
[4] C. Zhang, H. Li, X. Wang, et al., Cross-scene crowd counting via deep convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 833–841.
[5] D. Kong, D. Gray, H. Tao, Counting pedestrians in crowds using viewpoint invariant training, British Machine Vision Conf. (BMVC), IEEE Computer Society, 2005.
[6] H. Idrees, I. Saleemi, C. Seibert, et al., Multi-source multi-scale counting in extremely dense crowd images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2547–2554.
[7] D. Conte, P. Foggia, G. Percannella, et al., A method based on the indirect approach for counting people in crowded scenes, 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance, 2010.
[8] W. Ouyang, X. Wang, Joint deep learning for pedestrian detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2056–2063.
[9] P. Buehler, M. Everingham, D.P. Huttenlocher, et al., Upper body detection and tracking in extended signing sequences, Int. J. Comput. Vision 95 (2) (2011) 180–197.
[10] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1, IEEE, 2005, pp. 886–893.
[11] C. Zeng, H. Ma, Robust head-shoulder detection by pca-based multilevel hog-lbp detector for people counting, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 2069–2072.
[12] S. Wang, J. Zhang, Z. Miao, A new edge feature for head-shoulder detection, in: 2013 IEEE International Conference on Image Processing, IEEE, 2013, pp. 2822–2826.
[13] L. Chen, H. Wu, S. Zhao, et al., Head-shoulder detection using joint HOG features for people counting and video surveillance in library, in: Electronics, Computer and Applications, 2014 IEEE Workshop on, IEEE, 2014, pp. 429–432.
[14] D. Merad, K.E. Aziz, N. Thome, Fast people counting using head detection from skeleton graph, in: Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on, IEEE, 2010, pp. 233–240.
[15] K. Aziz, D. Merad, R. Iguernaissi, et al., Head detection based on skeleton graph method for counting people in crowded environments, J. Electron. Imaging 25 (1) (2016) 013012-013012.
[16] T. Van Oosterhout, S. Bakkes, B.J.A. Kröse, Head detection in stereo data for people counting and segmentation, VISAPP (2011) 620–625.
[17] P. Dollar, C. Wojek, B. Schiele, et al., Pedestrian detection: an evaluation of the state of the art, IEEE Trans. Pattern Anal. Mach. Intell. 34 (4) (2012) 743–761.
[18] C. Gao, P. Li, Y. Zhang, et al., People counting based on head detection combining Adaboost and CNN in crowded surveillance environment, Neurocomputing (2016).
[19] Y. Zhang, D. Zhou, S. Chen, et al., Single-image crowd counting via multi-column convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 589–597.
[20] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.
[21] Y. Jia, E. Shelhamer, J. Donahue, et al., Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, ACM, 2014, pp. 675–678.
[22] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 3642–3649.

[23] P. Sermanet, D. Eigen, X. Zhang, et al., Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013.
[24] R. Girshick, J. Donahue, T. Darrell, et al., Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.
[25] D. Erhan, C. Szegedy, A. Toshev, et al., Scalable object detection using deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2147–2154.
[26] C. Szegedy, A. Toshev, D. Erhan, Deep neural networks for object detection, Adv. Neural Inf. Process. Syst. (2013) 2553–2561.
[27] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[28] K. Zhang, L. Zhang, Q. Liu, et al., Fast visual tracking via dense spatio-temporal context learning, in: European Conference on Computer Vision, Springer International Publishing, 2014, pp. 127–141.
[29] D.S. Lee, Effective Gaussian mixture learning for video background subtraction, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 827–832.
[30] R. Girshick, J. Donahue, T. Darrell, et al., Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[31] M.M. Cheng, Z. Zhang, W.Y. Lin, et al., BING: binarized normed gradients for objectness estimation at 300fps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3286–3293.
[32] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[33] J. Deng, W. Dong, R. Socher, et al., Imagenet: a large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.
[34] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015, pp. 448–456.
[35] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Physica-Verlag HD, 2010, pp. 177–186.
[36] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.
[37] G.E. Hinton, N. Srivastava, A. Krizhevsky, et al., Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.
[38] S.B. Aruoba, J. Fernández-Villaverde, A comparison of programming languages in economics, Natl. Bureau Econ. Res. (2014).
[39] C. Cullinan, C. Wyant, T. Frattesi, et al., Computing Performance Benchmarks among CPU, GPU, and FPGA, 2013. Internet: www.wpi.edu/Pubs/E-project/Available/E-project-030212-123508/unrestricted/Benchmarking Final.
[40] N.D. Lane, P. Georgiev, Can deep learning revolutionize mobile sensing? in: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, ACM, 2015, pp. 117–122.
[41] R.K. Srivastava, K. Greff, J. Schmidhuber, Highway networks. arXiv preprint arXiv:1505.00387, 2015.
[42] L. Lu, Sequence training and adaptation of highway deep neural vetworks. arXiv preprint arXiv:1607.01963, 2016.
[43] K. He, X. Zhang, S. Ren, et al., Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, ACM, 2016, pp. 770–778.
[44] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[45] G. Huang, G.B. Huang, S. Song, et al., Trends in extreme learning machines: a review, Neural Netw. 61 (2015) 32–48.