Vehicle Type Classification Using a Semisupervised Convolutional Neural Network

Zhen Dong, Yuwei Wu, Mingtao Pei, and Yunde Jia, *Member, IEEE*

***Abstract*—In this paper, we propose a vehicle type classification method using a semisupervised convolutional neural network from vehicle frontal-view images. In order to capture rich and discriminative information of vehicles, we introduce sparse Laplacian filter learning to obtain the filters of the network with large amounts of unlabeled data. Serving as the output layer of the network, the softmax classifier is trained by multitask learning with small amounts of labeled data. For a given vehicle image, the network can provide the probability of each type to which the vehicle belongs. Unlike traditional methods by using handcrafted visual features, our method is able to automatically learn good features for the classification task. The learned features are discriminative enough to work well in complex scenes. We build the challenging BIT-Vehicle dataset, including 9850 high-resolution vehicle frontal-view images. Experimental results on our own dataset and a public dataset demonstrate the effectiveness of the proposed method.**

***Index Terms*— Feature learning, filter learning, multitask learning, neural network, vehicle type classification.**

I. Introduction

VEHICLE type classification is one of the essential parts in intelligent traffic system, and has a wide range of applications including traffic flow statistics, intelligent parking systems, and vehicle type detection. Existing methods are commonly based on ultrasonic, magnetic induction, and cameras. With the extensive use of traffic surveillance cameras, image-based methods have received significant attention in computer vision community. So far, numerous image-based methods have been proposed, and they roughly fall into two categories: model-based methods and appearance-based methods. Model-based methods [1]–[4] compute the vehicle's 3D parameters such as length, width, and height to recover the 3D model of the vehicle. Appearance-based methods [5]–[9] extract appearance features (e.g., SIFT [10], Sobel edges [11]) to represent the vehicle for classification. Most of these methods are based on vehicle side view images. Currently, large numbers of vehicle frontal view images are captured by traffic surveillance cameras, so we focus on vehicle type classification from vehicle frontal view images.

There has been less effort on methods based on vehicle frontal view images. Petrovic and Cootes [12] modeled the vehicle appearance from vehicle frontal view images by extracting many features, such as Sobel edge response, edge orientation, direct normalized gradients, locally normalized gradients, and Harris corner response. Negri *et al.* [13] presented a voting algorithm based on oriented-contour points for their multiclass vehicle type recognition system. Psyllos *et al.* [14] used SIFT features to recognize the logo, manufacturer, and model of a vehicle. Zhang [15] fused the PHOG feature and the Gabor transform feature to represent the vehicle and proposed a cascade classifier scheme to recognize the type of the vehicle. Peng *et al.* [16] represented a vehicle by license plate color, vehicle front width, and type probabilities for vehicle type classification. However, these methods use multiple hand-crafted features which might not be discriminative enough in complex scenes.

In this paper, we propose a novel framework of vehicle type classification from vehicle frontal view images using a convolutional neural network. The convolutional neural network is a multi-layer feed-forward neural network which is biologically-inspired. Unlike traditional methods by using hand-crafted features, the convolutional neural network is able to automatically learn multiple stages of invariant features for the specific task [17]. It has been used to learn good features in face detection [18], [19], facial point detection [20], pedestrian detection [21], human attribute inference [22], image quality assessment [23], image classification [24], and video classification [25].

The convolutional neural network in our method takes an original vehicle image as the input and outputs the probability of each vehicle type to which the vehicle belongs. The network contains two stages, and each stage consists of the convolutional layer, the absolute rectification layer, the local contrast normalization layer, the average pooling layer, and the subsampling layer. The convolutional layer computes the convolutions between the input and a set of filters (filter bank), and provide a nonlinear representation of the input signal by using a point-wise nonlinear function. The absolute rectification layer and local contrast normalization layer perform a nonlinear transformation on the output of the convolutional layer. The average pooling layer and subsampling layer reduce the spatial resolution of the representation to achieve the robustness to both geometric distortions and small shifts.

Manuscript received June 27, 2014; revised October 21, 2014 and December 26, 2014; accepted February 2, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61203291, by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20121101120029, and by the Specialized Fund for Joint Building Program of Beijing Municipal Education Commission. The Associate Editor for this paper was P. Grisleri. *(Corresponding author: Yuwei Wu.)*

The authors are with Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: dongzhen@bit.edu.cn; wuyuwei@bit.edu.cn; peimt@bit.edu.cn; jiayunde@bit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2015.2402438

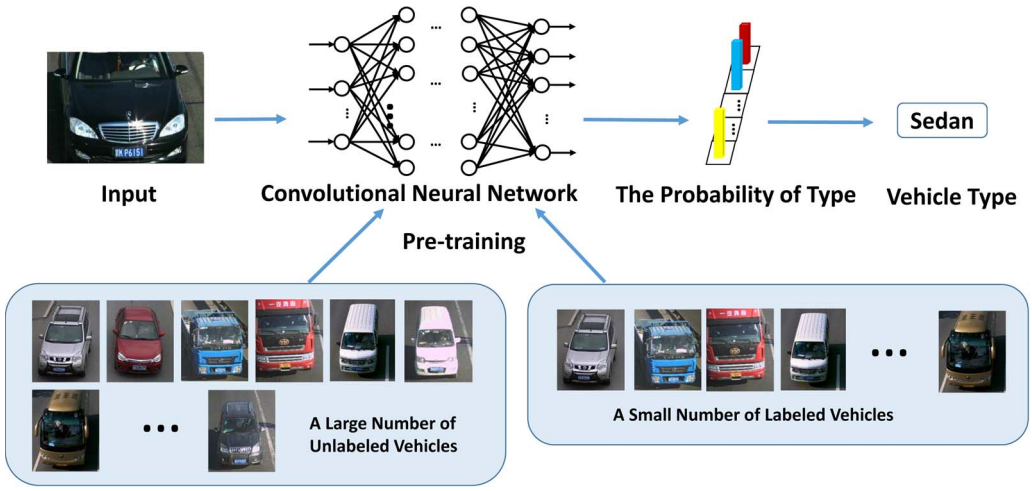


Fig. 1. The flowchart of the proposed method. Our convolutional neural network is semi-supervised. In order to capture rich and discriminative information of vehicles, the sparse Laplacian filter learning is employed to learn the filters of the network with a large number of unlabeled vehicles. The softmax classifier layer which is trained by multi-task learning with a small number of labeled vehicles is used as the output layer. For a given vehicle image, the network is able to automatically learns good features to represent the vehicle and outputs the probability of each type which the vehicle belongs to. The type of the vehicle can be predicted by picking the label which makes the probability achieving maximum.

Our network is semi-unsupervised, as shown in Fig. 1. The filter bank of the convolutional layer is learned in an unsupervised manner with large amounts of unlabeled data, and the parameters of the output layer are learned in a supervised manner with a certain amount of labeled data. Motivated by great success of unsupervised pre-training for multi-layer neural networks [26]–[29], we introduce the sparse Laplacian filter learning (SLFL), an unsupervised learning method, to learn the filter bank of the convolutional layer. Different from traditional sparsity constraints like $l0, l1$ or $l2$ norms, the SLFL uses the sparse function $\mathrm{sps}(\cdot)$ with properties of population sparsity, high dispersal, and lifetime sparsity to measure the sparsity of representations. During learning filters, the manifold assumption [30] is considered to ensure that similar input image patches have similar high-level representations. The learned filters are able to capture rich and discriminative information of vehicles for improving the classification performance.

We adopt a softmax classifier as the output layer to calculate the probability of each vehicle type. A supervised learning method is introduced to learn the parameters of the classifier. We observe that many vehicle types share large number of common appearance patterns. For example, the vehicles of "truck" and "minivan" have the similar parts layouts, and both of them have a hopper. The multi-task learning [31] is used to learn the shared common appearance patterns. The appearance pattern is regarded as a latent task, and the parameter of each type model is reconstructed by the linear combination of the latent tasks. The constraints of the latent tasks and combination coefficients are important to obtain a robust model for classification. To enhance the discriminative power of the latent task, we represent the important appearance patterns by employing the $l1$-norm. The $l1$-norm of each category's combination coefficient vector is able to reduce the noise of reconstruction.

The rest of the paper is organized as follows. In Section II, we show the architecture of the convolutional neural network and its implementation. In Section III, the learning methods of the network parameters are described, including learning the filer bank and the parameters of the softmax layer. Experimental results and discussions are reported in Sections IV and V concludes this paper.

II. Architecture of the Convolutional Neural Network

The architecture of our convolutional neural network is shown in Fig. 2. The network contains two stages which generate low-level local features and high-level global features, respectively. The high-level global features provide holistic descriptions of vehicles, and the low-level features aim to characterize vehicle parts precisely. In order to take full advantage of both global features and local features, the network is with layer-skipping which integrates the features learned in both the 1st and the 2nd stage for classification. There are five layers in each stage, i.e., the convolutional layer, the absolute value rectification layer, the local contrast normalization layer, the average pooling layer, and the subsampling layer. In the convolutional layer, the sparse Laplacian filter learning (SLFL) provides effective filters for the network. The absolute rectification layer and local contrast normalization layer provide a nonlinear transformation for the output of the convolutional layer. The pooling and subsampling layers use the average pooling operator to reduce the spatial resolution of the representation. The representations can thus be robust to geometric distortions and small shifts. We employ a softmax classifier as the output layer of the network to compute the probability of each vehicle type. For simplicity, we use $\boldsymbol{x}$ and $\boldsymbol{y}$ to represent the input and output of each layer, respectively. They are both 3D arrays where $\boldsymbol{x}$ are with the size of $s1 \times s2 \times s3$ and $\boldsymbol{y}$ with the size of $t1 \times t2 \times t3$.

A. Convolutional Layer

In the convolutional layer, convolutions between the input and a series of filters are first computed. An element-wise nonlinear activation function is then executed on the convolutions.

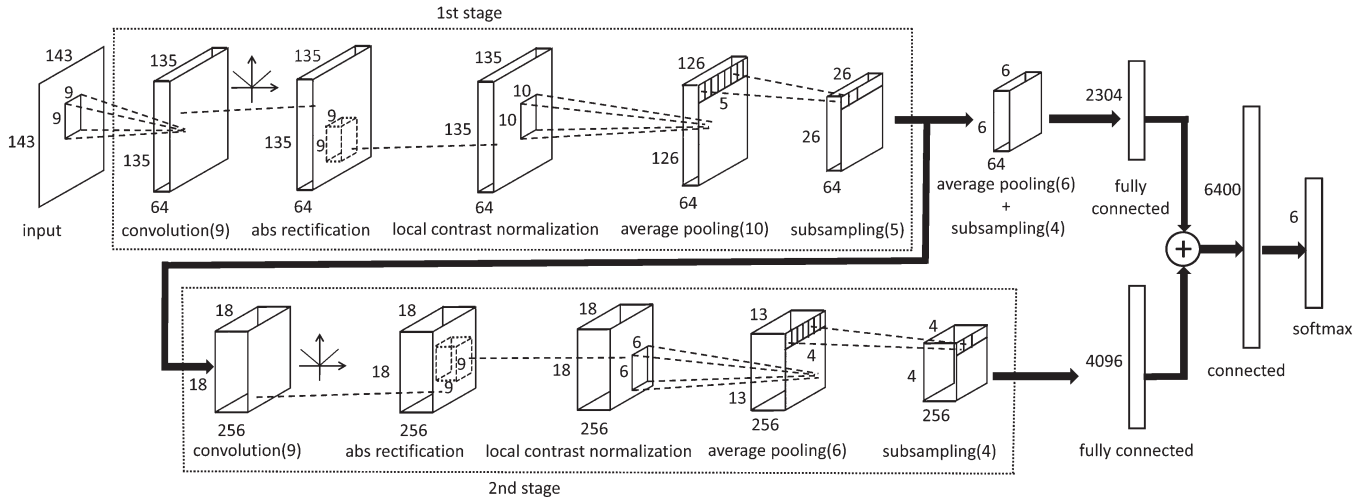


Fig. 2. The architecture of our semi-supervised convolutional neural network. The network contains two stages. Each stage consists of the convolutional layer, the absolute rectification layer, the local contrast normalization layer, the average pooling layer, and the subsampling layer. The number in brackets behind "convolution" is the size of the filters. The number in brackets behind "average pooling" is the size of the average filter, and the number in brackets behind "subsampling" reflects the step of subsampling. The numbers around cuboids describe the size of 3D feature arrays. The network is with layer-skipping which integrates the features learned in both 1st and 2nd stage together for classification. The softmax classifier is applied as the output layer to compute the type probability of the input vehicle.

The layer provides a non-linear mapping from the low level image representation to high level semantic understanding, which simulates the "simple cells" in the standard models of the visual cortex [32], [33]. The input $\boldsymbol{x}$ is a 3D array with the size of $s1 \times s2 \times s3$, where $s3$ is the number of 2D feature maps, and $s1 \times s2$ is the size of the 2D feature map which is represented by $\boldsymbol{x}_i$. The output $\boldsymbol{y}$ is also a 3D array whose size is $t1 \times t2 \times t3$. Similar to the 2D feature map $\boldsymbol{x}_i$ of the input, $\boldsymbol{y}_j$ is defined as the j-th 2D feature map of the output. The element-wise sigmoid function $\mathrm{sig}(\cdot)$ is chosen as the non-linear activation function. Hence, $\boldsymbol{y}_j$ is computed by

$$\boldsymbol{y}_j = \mathrm{sig}\left(\sum_i \boldsymbol{k}_{ij} \otimes \boldsymbol{x}_i\right), \tag{1}$$

where $\otimes$ denotes the convolution operation, and $\boldsymbol{k}_{ij}$ is a 2D filter learned by the sparse Laplacian filter learning described in Section III-A. Suppose that the filter size is $l1 \times l2$, we have $t1 = s1 - l1 + 1$ and $t2 = s2 - l2 + 1$ due to the board effects. The size of input 2D feature map is an important factor. A large size is beneficial for learning good features of vehicles, but the computation time cost will be high. A small size saves time, but may lose too much information, which leads to low classification accuracy. As shown in Fig. 2, in the 1st stage of the network, the size of the input 2D feature map is set as 143×143 to take a balance between the computation time cost and the accuracy. The size of output 2D feature map is 135×135 since the filters are with the size of 9×9 which is set according to the size of the input 2D feature map.

B. Absolute Value Rectification Layer

In this module, all the elements are passed into the absolute value rectification function

$$\boldsymbol{y}_{i,j,k} = |\boldsymbol{x}_{i,j,k}|, \tag{2}$$

where $\boldsymbol{x}_{i,j,k}$ and $\boldsymbol{y}_{i,j,k}$ represent each element of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. The absolute value rectification layer is inspired by the fact that the relationship between two items in real world is always positive or zero, but not negative. It is clear that the sizes of the input and the output of the absolute value rectification layer are the same.

C. Local Contrast Normalization Layer

The purpose of the local contrast normalization layer is to enforce local competitions between one neuron and its neighbors, which is motivated by the computational neuroscience [34], [35]. The neighbors include nearby neurons in the same feature map and the ones at the same 2D location in different feature maps. To do this, two normalization operations are performed, i.e., subtractive and divisive. For the element $\boldsymbol{x}_{i,j,k}$ in the input 3D array size of $s1 \times s2 \times s3$, the subtractive normalization operator is given by

$$\boldsymbol{z}_{i,j,k} = \boldsymbol{x}_{i,j,k} - \sum_{p=-4}^{4}\sum_{q=-4}^{4}\sum_{r=1}^{s3} \boldsymbol{\omega}_{p,q}\boldsymbol{x}_{i+p,j+q,r}, \tag{3}$$

where $\boldsymbol{\omega}_{p,q}$ is a normalized Gaussian filter with the size of 9×9, $\boldsymbol{z}$ is the output of the subtractive normalization and the input of the divisive normalization. The divisive normalization operator is defined as

$$\boldsymbol{y}_{i,j,k} = \frac{\boldsymbol{z}_{i,j,k}}{max\left(M, M(i,j)\right)}, \tag{4}$$

where

$$M(i,j) = \sqrt{\sum_{p=-4}^{4}\sum_{q=-4}^{4}\sum_{r=1}^{s3} \boldsymbol{\omega}_{p,q}\boldsymbol{z}_{i+p,j+q,r}^2}, \tag{5}$$

$$M = \left(\sum_{i=1}^{s1}\sum_{j=1}^{s2} M(i,j)\right)/(s1 \times s2). \tag{6}$$

In both operations, the filtering by $\boldsymbol{\omega}_{p,q}$ is computed with the zero-padded edges. It is obvious that the size of the output is the same as the input in the local contrast normalization layer.

D. Average Pooling and Subsampling Layers

The pooling and subsampling layers aim to make the representation robust to both geometric distortions and small shifts. Their roles are essentially equivalent to the "complex cells" in the standard models of the visual cortex [32], [33]. We adopt the average pooling method in the pooling layer. The convolution between the 2D feature map and the average filter is formulated as

$$\boldsymbol{y}_{i,j,k} = \sum_{p,q} \boldsymbol{\alpha}_{p,q} \boldsymbol{x}_{i+p,j+q,k}, \tag{7}$$

where $\boldsymbol{\alpha}_{p,q} = 1/(f1 \times f2)$ is the average filter with the size of $f1 \times f2$, $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output 3D arrays of the average pooling layer, respectively.

The subsampling procedure is performed on the output of the average pooling layer with the rate of $p1$ horizontally and $p2$ vertically. Suppose that the input 2D feature maps of these two layers are size of $s1 \times s2$, and the size of the output 2D feature map is $t1 \times t2$, we have

$$\begin{aligned} t1 &= \left\lfloor \frac{s1 - f1}{p1} \right\rfloor + 1, \\ t2 &= \left\lfloor \frac{s2 - f2}{p2} \right\rfloor + 1, \end{aligned} \tag{8}$$

where $\lfloor \delta \rfloor$ denotes the maximum integer less or equal than δ. For example, the input 3D array of the pooling layer in the 1st stage is with the width and height of $s1 = s2 = 135$. The appropriate average filter size and subsampling rate are set for simplicity. The average filter size is $f1 = f2 = 10$, and the subsampling rates are 5 in both the horizontal direction and the vertical direction. Therefore, the size of the output 2D array is $t1 = t2 = \lfloor (135 - 10)/5 \rfloor + 1 = 26$.

E. Softmax Classifier Layer

In order to calculate the probability of each vehicle type, the softmax classifier is employed as the output layer of the convolutional neural network. As shown in Fig. 2, the input of the softmax classifier layer is the feature vector learned by previous layers, and the output is the type probability vector. A linear function is applied to model the relationship between the feature and the probability distribution of the vehicle type

$$\boldsymbol{v} = W^\top \boldsymbol{x} + \boldsymbol{b}, \tag{9}$$

where $\boldsymbol{x} \in \mathbb{R}^{D \times 1}$ represents the input feature, $\boldsymbol{v} \in \mathbb{R}^{C \times 1}$ is a intermediate variable for describing the distribution, and C is the number of vehicle types. For simplicity, Eq. (9) is rewritten as

$$\boldsymbol{v} = W^\top \boldsymbol{x} + \boldsymbol{b} = [W^\top \ \boldsymbol{b}] \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = \mathcal{W}^\top \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}, \tag{10}$$

where $\mathcal{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_C] \in \mathbb{R}^{(D+1) \times C}$, and each column of $\mathcal{W}$ is the corresponding vehicle type model parameter. Because the probability has the properties of nonnegativity and unitarity, $\boldsymbol{v}$ is normalized as

$$\begin{aligned} \boldsymbol{y}_i &= \frac{1}{V} e^{\boldsymbol{v}_i}, i = 1, 2, \cdots, C, \\ V &= \sum_{i=1}^{C} e^{\boldsymbol{v}_i}, \end{aligned} \tag{11}$$

where $\boldsymbol{v}_i$ is the i-th element of $\boldsymbol{v}$, and $\boldsymbol{y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_C]^\top$ is the output of the softmax classifier layer. The parameter $\mathcal{W}$ can be learned by the multi-task learning described in Section III-B.

III. Parameters Learning

As discussed in Section II, two kinds of parameters of the network should be learned, i.e., the filter bank of the convolutional layer and the parameter of the softmax classifier. In this section, we elaborate how to learn these two parameters.

A. Sparse Laplacian Filter Learning

We introduce the sparse Laplacian filter learning (SLFL), an unsupervised learning method, to learn the filter bank of the convolutional neural network. Define a data matrix as $\mathcal{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n] \in \mathbb{R}^{d \times n}$, where the columns are data points. Our goal is to learn the filter bank $\mathcal{K} \in \mathbb{R}^{d \times t}$ which consists of t filters. Using this filter bank, the input data points $\mathcal{U}$ can be mapped to sparse representations $\mathcal{A}$. The nonlinear map function is given by

$$\mathcal{A} = \mathrm{sig}(\mathcal{K}^\top \mathcal{U}), \tag{12}$$

where $\mathrm{sig}(\cdot)$ is the element-wise sigmoid function which is commonly used as the activation function of the neural network. $\mathcal{A} \in \mathbb{R}^{t \times n}$ is the feature distribution matrix over $\mathcal{U}$, where the row is a feature and the column is an example. The element $\mathcal{A}_{i,j}$ represents the activation of the i-th feature on the j-th example of $\mathcal{A}$. We formulate the sparse Laplacian filter learning as

$$\begin{aligned} \min_{\mathcal{B},\mathcal{A},\mathcal{K}} \|\mathcal{U} - \mathcal{B}\mathcal{A}\|_F^2 + \alpha \mathrm{sps}(\mathcal{A}) \\ + \ \beta tr(\mathcal{A}\mathcal{L}\mathcal{A}^\top) + \gamma \|\mathcal{A} - \mathrm{sig}(\mathcal{K}^\top \mathcal{U})\|_F^2, \end{aligned} \tag{13}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of the matrix, $tr(\cdot)$ represents the trace of a matrix, $\mathcal{L}$ is the Laplacian matrix, α, β, and γ are the regularized parameters. Similar to the sparse coding, the first term pursues accurate reconstruction by the dictionary $\mathcal{B}$, in other words, each data point $\boldsymbol{u}_i$ can be linearly represented by the bases of the dictionary $\mathcal{B}$, meanwhile keeping the reconstruction error as small as possible.

The $\mathrm{sps}(\cdot)$ function in Eq. (13) is optimized for the sparsity in the feature distribution [36]. It avoids modeling the data distribution explicitly, which gives rise to a simple formulation and permits the effectiveness of learning. Let $\mathcal{A}_{(i,\triangle)} \in \mathbb{R}^{1 \times n} (i = 1, 2, \cdots, t)$ be the i-th row of $\mathcal{A}$, and $\mathcal{A}_{(\triangle,j)} \in \mathbb{R}^{t \times 1} (j = 1, 2, \cdots, n)$ the j-th column of $\mathcal{A}$. The $\mathrm{sps}(\cdot)$ function is computed in three steps: normalizing the feature distribution matrix by rows, normalizing the feature distribution matrix

by columns, and summing up the absolute values of all entries. In the first step, each feature is divided by its $\ell 2$-norm across all examples, i.e., $\widetilde{\mathcal{A}}_{(i,\triangle)} = \mathcal{A}_{(i,\triangle)}/\|\mathcal{A}_{(i,\triangle)}\|_2$. In this way, the feature is equally active. In the second step, each example is divided by its $\ell 2$-norm across all features, $\hat{\mathcal{A}}_{(\triangle,j)} = \widetilde{\mathcal{A}}_{(\triangle,j)}/\|\widetilde{\mathcal{A}}_{(\triangle,j)}\|_2$, to ensure that all examples lie on the unit $\ell 2$-ball. In the third step, we sum up the absolute values of all the entries in $\hat{\mathcal{A}}$. The $\mathrm{sps}(\cdot)$ function is given by

$$\begin{aligned}\min_{\mathcal{K}} \|\hat{\mathcal{A}}\|_* &= \sum_{i=1}^{t}\sum_{j=1}^{n} |\hat{\mathcal{A}}_{i,j}| \\ &= \sum_{j=1}^{n} \|\hat{\mathcal{A}}_{(\triangle,j)}\|_1 = \sum_{j=1}^{n} \left\| \frac{\widetilde{\mathcal{A}}_{(\triangle,j)}}{\|\widetilde{\mathcal{A}}_{(\triangle,j)}\|_2} \right\|_1, \end{aligned} \tag{14}$$

where $\|\mathcal{M}\|_*$ denotes summing up the absolute values of all the elements in the matrix $\mathcal{M}$. The minimization of the sparse function makes the feature distribution $\mathcal{A}$ have three properties, i.e., population sparsity, high dispersal, and lifetime sparsity [37], [38].

Population Sparsity: The population sparsity requires that the example should only have a few active (non-zero) features, and it is considered to be an efficient coding method in early vision cortex. The term $\|\hat{\mathcal{A}}_{(\triangle,j)}\|_1 = \left\| \frac{\|\widetilde{\mathcal{A}}_{(\triangle,j)}}{\|\widetilde{\mathcal{A}}_{(\triangle,j)}\|_2} \right\|_1$ in Eq.(14) reflects the population sparsity property of the features on the j-th example. Since $\hat{\mathcal{A}}_{(\triangle,j)}$ has been constrained to lie on the unit $\ell 2$-ball, the objective function is minimized when the features are sparse. In other words, the objective tends to place the examples close to feature axes, and the example which has similar values on the features would have a high penalty.

High Dispersal: The high dispersal property denotes that the statistics of the features should be similar, which implies that the contributions of all features are similar. Here, the statistics are taken as the mean squared activations by averaging the squared values in the feature matrix across all the examples

$$\mathcal{T}_i = \frac{1}{n}\sum_{j=1}^{n} \mathcal{A}_{ij}^2 = \frac{1}{n}\|\mathcal{A}_{(i,\triangle)}\|_2^2. \tag{15}$$

Each feature is divided by its $\ell 2$-norm across all examples in the first step of computing the objective function, which makes the features equally active. Therefore, the objective function is optimized for high dispersal.

Lifetime Sparsity: The property that the feature should be active only on a few examples is called lifetime sparsity. This property guarantees that the feature is discriminative enough to distinguish different examples. Specifically, each row of the feature distribution matrix should only have a few active (non-zero) entries. In the sparse filtering algorithm, the lifetime sparsity property is ensured by the population sparsity property and the high dispersal property. The feature distribution matrix should have a great many non-active (zero) elements due to the population sparsity property. These non-active elements could not be placed in a few specific rows, otherwise it would be against the high dispersal property. Therefore, the feature would have a significant number of non-active elements and thus be lifetime sparse.

The third term of Eq. (13) incorporates the manifold assumption into the objective function. The manifold assumption implies that close-by data points tend to have similar representations and distant ones are less likely to take similar representations. This can be achieved by approximating the structure of the manifold with a graph. Each point of the graph represents a data point $\boldsymbol{x}_i$, and the edge weight matrix $\mathcal{R}$ is defined as

$$\mathcal{R}_{ij} = \begin{cases} \frac{\boldsymbol{u}_i^\top \boldsymbol{u}_j}{\|\boldsymbol{u}_i\|\|\boldsymbol{u}_j\|} & \text{if } \boldsymbol{u}_i \in N_\varepsilon(\boldsymbol{u}_j) \text{ or } \boldsymbol{u}_j \in N_\varepsilon(\boldsymbol{u}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where $N_\varepsilon(\boldsymbol{u}_i)$ represents the set of ε nearest neighbors of $\boldsymbol{u}_i$. The edge weight matrix satisfies that large values $\mathcal{R}_{ij}$ is corresponding to nearby data points. The manifold assumption is formulated as the minimization of

$$\frac{1}{2}\sum_{i,j=1}^{n} \|\mathcal{A}_{(\triangle,i)} - \mathcal{A}_{(\triangle,j)}\|^2 = Tr(\mathcal{A}\mathcal{L}\mathcal{A}^\top), \tag{17}$$

where $\mathcal{L} = \mathcal{D} - \mathcal{R}$ is the Laplacian matrix, and $\mathcal{D}$ is a diagonal matrix whose elements are column (or row) sums of $\mathcal{R}$.

The last term of Eq. (13) pursuits the minimal error between $\mathcal{A}$ and the nonlinear mapping of $\mathcal{U}$. The term is added into the objective function to optimize $\mathcal{B}, \mathcal{A}$, and $\mathcal{K}$ jointly. The optimization procedure contains two alternating steps.

STEP 1: Keeping parameters $\mathcal{B}$ and $\mathcal{K}$ fixed, learn the representation $\mathcal{A}$ by solving the optimization problem:

$$\begin{aligned}\min_{\mathcal{A}} \|\mathcal{U} - \mathcal{B}\mathcal{A}\|_F^2 + \alpha\,\mathrm{sps}(\mathcal{A}) \\ + \beta Tr(\mathcal{A}\mathcal{L}\mathcal{A}^\top) + \gamma\|\mathcal{A} - \mathrm{sig}(\mathcal{K}^\top\mathcal{U})\|_F^2. \end{aligned} \tag{18}$$

In our implementation, the L-BFGS optimization method [39] is used. The gradient of the objective function in Eq. (18) with respect to $\mathcal{A}$ is easy to solve. The gradient of $\mathrm{sps}(\cdot)$ function is computed by the back propagation algorithm. As the $\mathrm{sps}(\cdot)$ contains absolute value operators which are nondifferentiable, we ignore the absolute value operators when calculating the gradient to get an approximation.

STEP 2: With the optimal value of $\mathcal{A}$ from STEP 1, minimize Eq. (13) with respect to $\mathcal{B}$ and $\mathcal{K}$. The optimization problem is rewritten as

$$\min_{\mathcal{B},\mathcal{K}} \|\mathcal{U} - \mathcal{B}\mathcal{A}\|_F^2 + \gamma\|\mathcal{A} - \mathrm{sig}(\mathcal{K}^\top\mathcal{U})\|_F^2. \tag{19}$$

Because two terms of the objective function are not correlated, they can be solved independently. The optimal dictionary $\mathcal{B}$ can be achieved by minimizing

$$\min_{\mathcal{B}} \|\mathcal{U} - \mathcal{B}\mathcal{A}\|_F^2. \tag{20}$$

An analytical solution of $\mathcal{B}$ is obtained that $\mathcal{B} = \mathcal{U}\mathcal{A}^\top(\mathcal{A}\mathcal{A}^\top)^{-1}$. The columns of $\mathcal{B}$ are then re-scaled to unit norm to avoid trivial solutions that are due to the ambiguity of the linear reconstruction. Unlike the first term, $\mathcal{K}$ cannot be solved analytically due to the element wise function. Instead,

the L-BFGS optimization method [39] is used to minimize the second term with respect to $\mathcal{K}$:

$$\min_{\mathcal{K}} \left\| \mathcal{A} - \mathrm{sig}(\mathcal{K}^\top \mathcal{U}) \right\|_F^2 . \tag{21}$$

The overall optimization procedure of the sparse Laplacian filter learning is summarized in Algorithm 1, where the "convergence" is the value difference of the objective function in Eq. (12) smaller than a threshold, or the iterative times exceeds another threshold. Since the Step 2 and 3 of Algorithm 1 are both based on L-BFGS, the computation complexity of the algorithm is $O(kt^2(n^2+d^2))$ where k is the iterative times.

Algorithm 1: Sparse Laplacian Filter Learning

Input: data matrix $\mathcal{U} \in \mathbb{R}^{d\times n}$, Laplacian Matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$, parameters $\alpha, \beta, \gamma \in \mathbb{R}^+$.
Output: dictionary $\mathcal{B} \in \mathbb{R}^{d\times t}$, representations $\mathcal{A} \in \mathbb{R}^{t\times n}$, filter bank $\mathcal{K} \in \mathbb{R}^{d\times t}$

Step 1. Initialization
  initialize dictionary $\mathcal{B}$ by using k-means method on $\mathcal{U}$;
  normalize each column of $\mathcal{B}$ to unit norm;
  $\mathcal{A} = (\mathcal{B}^\top \mathcal{B})^{-1} \mathcal{B}^\top \mathcal{U}$;
  $\mathcal{K} = (\mathcal{U}\mathcal{U}^\top)^{-1} \mathcal{U}\mathcal{A}^\top$;
repeat
  Step 2. Optimize $\mathcal{A}$ with fixed $\mathcal{B}$ and $\mathcal{K}$
    solve Eq.(18) to obtain $\mathcal{A}$ by L-BFGS;
  Step 3. Keeping $\mathcal{A}$ fixed, optimize $\mathcal{B}$ and $\mathcal{K}$
    $\mathcal{B} = \mathcal{U}\mathcal{A}^\top(\mathcal{A}\mathcal{A}^\top)^{-1}$;
    normalize each column of $\mathcal{B}$ to unit norm;
    solve Eq.(21) to obtain $\mathcal{K}$ by L-BFGS;
until *Convergence*;
Step 4. output $\mathcal{B}, \mathcal{A}$ and $\mathcal{K}$;

B. Multi-Task Learning

We provide a supervised learning procedure for the parameters of the softmax layer. The learning method is based on the observation that many vehicle types share a great many common appearance patterns. We use the multi-task learning method [31] to learn the shared knowledge between different vehicle types. The shared knowledge is corresponding to the latent tasks, and the model of each vehicle type can be combined by the latent tasks.

Specifically, each column of $\mathcal{W}$ is a vehicle type classifier and can be represented by the linear combination of the latent tasks. Define $\mathcal{T} \in \mathbb{R}^{(D+1)\times K}$ as the shared latent task matrix with each column characterizing a latent task, and K is the number of latent tasks. The linear combination weight matrix is defined as $\mathcal{C} \in \mathbb{R}^{K\times C}$ with each column representing the combination coefficients of the corresponding vehicle type. We thus have

$$\mathcal{W} = \mathcal{T}\mathcal{C}. \tag{22}$$

We will learn $\mathcal{T}$ and $\mathcal{C}$ simultaneously instead of learning $\mathcal{W}$ directly.

Denote the training samples as $\{(\boldsymbol{x}^{(i)}, \boldsymbol{d}^{(i)}) | i = 1, 2, \cdots, N\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{(D+1)\times 1}$ is the input feature vector (with the additional constant dimension), $\boldsymbol{d}^{(i)}$ is the probability distribution of the type of $\boldsymbol{x}^{(i)}$. If $\boldsymbol{x}^{(i)}$ belongs to the j-th type $(1 \le j \le C)$, the j-th element of $\boldsymbol{d}^{(i)}$ will be 1 and others will be 0. In order to learn $\mathcal{T}$ and $\mathcal{C}$, we introduce the Kullback-Leibler (KL) divergence as the optimization principle

$$\begin{aligned}
&\min_{\mathcal{T},\mathcal{C}} \sum_{i=1}^{N} KL(\boldsymbol{d}^{(i)} \| \boldsymbol{y}^{(i)}) \\
&= \min_{\mathcal{T},\mathcal{C}} \sum_{i=1}^{N} \Big(\sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \frac{1}{\boldsymbol{y}_j^{(i)}} - \sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \frac{1}{\boldsymbol{d}_j^{(i)}} \Big) \\
&= \min_{\mathcal{T},\mathcal{C}} - \sum_{i=1}^{N} \sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)},
\end{aligned} \tag{23}$$

where $\boldsymbol{d}_j^{(i)}$ and $\boldsymbol{y}_j^{(i)}$ represent the j-th element of $\boldsymbol{d}^{(i)}$ and $\boldsymbol{y}^{(i)}$, respectively.

The constraints to $\mathcal{T}$ and $\mathcal{C}$ are also very important to learn a robust model for vehicle type classification. Discriminative information may be lost if vehicle types share too much holistic information. We expect that the latent tasks focus on the basic visual patterns that can be shared by vehicle types. To achieve this goal, the $\ell 1$-norm of the latent task is employed. We assume that the vehicle type model can be reconstructed by only a small number of latent tasks. Latent tasks are thus shared only among related vehicle types and hold high discriminative power. This can be achieved by minimizing the $\ell 1$-norm of each row of $\mathcal{C}$. The Frobenius-norm regularization of $\mathcal{T}$ is used to avoid overfitting. Taking these three constraints into account, the objective function for learning $\mathcal{T}$ and $\mathcal{C}$ is formulated as

$$\begin{aligned}
\min_{\mathcal{T},\mathcal{C}} - \sum_{i=1}^{N} \sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)} + \lambda \|\mathcal{T}\|_F^2 \\
+ \mu \|\mathcal{T}\|_* + \eta \|\mathcal{C}\|_*,
\end{aligned} \tag{24}$$

where $\|\cdot\|_*$ denotes summing up the absolute values of all the elements in the matrix.

The objective function of Eq. (24) is not jointly convex in $\mathcal{T}$ and $\mathcal{C}$, but it is convex in $\mathcal{C}$ with fixed $\mathcal{T}$ and convex in $\mathcal{T}$ with fixed $\mathcal{C}$. Therefore, we adopt the alternating optimization strategy to solve Eq.(24). The two steps of the optimization method are as follows.

STEP 1: With fixed $\mathcal{T}$, the optimal combination weight matrix $\mathcal{C}$ can be obtained by solving

$$\min_{\mathcal{C}} - \sum_{i=1}^{N} \sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)} + \eta \|\mathcal{C}\|_*. \tag{25}$$

The objective function is not smooth with respect to $\mathcal{C}$ as the existence of $\|\cdot\|_*$. Fortunately, the first term is a differentiable convex function, and the second term is convex but non-smooth. The accelerated proximal gradient method (APG) [40] is able to solve the optimization problem. For simplicity, we define

$$\begin{aligned}
f(\mathcal{C}) &= - \sum_{i=1}^{N} \sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)}, \\
g(\mathcal{C}) &= \eta \|\mathcal{C}\|_*.
\end{aligned} \tag{26}$$

Following the update scheme in [40], the APG uses the linear combination of previous two points $\mathcal{C}_{i-1}$ and $\mathcal{C}_{i-2}$ as the next

search point $\mathcal{C}_i$:

$$\mathcal{C}_i = \frac{r_{i-1}+r_{i-2}-1}{r_{i-1}}\mathcal{C}_{i-1} - \frac{r_{i-2}-1}{r_{i-1}}\mathcal{C}_{i-2},$$
$$\mathcal{C}_i = h\left(\mathcal{C}_i - \frac{1}{\xi}\nabla_{\mathcal{C}} f(\mathcal{C}_i); \frac{\eta}{\xi}\right), \tag{27}$$

where ξ is the Lipschitz constant calculated by the backtracking search method, r is initialized as 1 and updated as $r_i = (1+\sqrt{1+4r_{i-1}^2})/2$, and $h(x;\alpha) = \max(|x|-\alpha, 0)\,\mathrm{sgn}(x)$ is the shrinkage operator.

STEP 2: Keeping $\mathcal{C}$ fixed, the optimal latent task matrix $\mathcal{T}$ is learned by solving

$$\min_{\mathcal{T}} -\sum_{i=1}^{N}\sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)} + \lambda\|\mathcal{T}\|_F^2 + \mu\|\mathcal{T}\|_*. \tag{28}$$

Similar to STEP 1, Eq. (28) is also solved by using the APG algorithm [40], where $f(\mathcal{T})$ and $g(\mathcal{T})$ are defined as

$$f(\mathcal{T}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} \boldsymbol{d}_j^{(i)} \ln \boldsymbol{y}_j^{(i)} + \lambda\|\mathcal{T}\|_F^2,$$
$$g(\mathcal{T}) = \mu\|\mathcal{T}\|_*. \tag{29}$$

The overall algorithm for learning $\mathcal{W}$ is summarized in Algorithm 2. The algorithm is converged when the value difference of the objective function in Eq. (24) is under a small threshold. Since the second and third step of Algorithm 2 are both based on APG, their convergence rates are $O(k_1^{-2})$ and $O(k_2^{-2})$ where k_1 and k_2 are the iterative times of the two steps, respectively.

Obtaining the optimized parameters, we calculate the type probability of the test vehicle image by using the convolutional neural network. The type of the test vehicle can be predicted by picking the label which makes the probability achieving maximum.

Algorithm 2: Multi-task learning algorithm for solving $\mathcal{W}$ of the softmax classifer

Input: training data $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \in \mathbb{R}^{D+C} | i = 1, 2, \cdots, N\}$, parameters $\lambda, \mu, \eta \in \mathbb{R}^+$, threshold $\epsilon \in \mathbb{R}$.
Output: $\mathcal{W} \in \mathbb{R}^{D\times C}$ of the softmax layer

Step 1. Initialization
train a logistic regression classifier for each type using its own data;
put all these initial logistic parameters together to initialize $\mathcal{W}$;
compute the SVD decomposition of $\mathcal{W}$: $\mathcal{W} = \Gamma\Sigma\Lambda$;
initialize $\mathcal{T} \in \mathbb{R}^{D\times K}$ with K columns of Γ corresponding to the top K singular values;
initialize $\mathcal{C} \in \mathbb{R}^{K\times C}$ with random values;
repeat
  Step 2. Optimize $\mathcal{C}$ with fixed $\mathcal{T}$
  $r_0 = 1, r_1 = (1+\sqrt{5})/2, i = 2, \mathcal{C}_1 = \mathcal{C}$;
  repeat
    $\mathcal{C}_i = \frac{r_{i-1}+r_{i-2}-1}{r_{i-1}}\mathcal{C}_{i-1} - \frac{r_{i-2}-1}{r_{i-1}}\mathcal{C}_{i-2}$;
    $\mathcal{C}_i = h\big(\mathcal{C}_i - \frac{1}{\xi}\nabla_{\mathcal{C}} f(\mathcal{C}_i); \frac{\eta}{\xi}\big)$;
    $r_i = (1+\sqrt{1+4r_{i-1}^2})/2$
    $i = i+1$;
  until $|\mathcal{C}_{i-1} - \mathcal{C}_{i-2}| < \epsilon$;
  get the optimal $\mathcal{C}$ with current $\mathcal{T}$;
  Step 3. Optimize $\mathcal{T}$ with fixed $\mathcal{C}$
  $r_0 = 1, r_1 = (1+\sqrt{5})/2, i = 2, \mathcal{T}_1 = {}_1\mathcal{T}$;
  repeat
    $\mathcal{T}_i = \frac{r_{i-1}+r_{i-2}-1}{r_{i-1}}\mathcal{T}_{i-1} - \frac{r_{i-2}-1}{r_{i-1}}\mathcal{T}_{i-2}$;
    $\mathcal{T}_i = h\big(\mathcal{T}_i - \frac{1}{\xi}\nabla_{\mathcal{T}} f(\mathcal{T}_i); \frac{\mu}{\xi}\big)$;
    $r_i = (1+\sqrt{1+4r_{i-1}^2})/2$
    $i = i+1$;
  until $|\mathcal{T}_{i-1} - \mathcal{T}_{i-2}| < \epsilon$;
  get the optimal $\mathcal{T}$ with current $\mathcal{C}$;
until *Convergence*;
Step 4. output $\mathcal{W} = \mathcal{T}\mathcal{C}$;

IV. Experiments

A. Datasets, Settings, and Preprocessing

We constructed a complex and challenging vehicle dataset called BIT-Vehicle Dataset[1] which includes 9,850 vehicle images to test the proposed method. The proportion of nightlight images in the whole dataset is about 10%. Fig. 3 shows the example images of the dataset, there are images with sizes of 1600 × 1200 and 1920 × 1080 captured from two cameras at different time and places. The images contain changes in the illumination condition, the scale, the surface color of vehicles, and the viewpoint. The top or bottom parts of some vehicles are not included in the images due to the capturing delay and the size of the vehicle. As shown in Fig. 3, there may be one or two vehicles in one image, so the location of each vehicle is pre-annotated. The dataset can also be used for evaluating the performance of vehicle detection. All vehicles in the dataset

Fig. 3. The example images of BIT-Vehicle Dataset. All vehicles in our dataset fall into 6 types: Bus, Microbus, Minivan, Sedan, SUV, and Truck.

are divided into six categories: Bus, Microbus, Minivan, Sedan, SUV, and Truck. The numbers of vehicles per vehicle type are 558, 883, 476, 5,922, 1,392, and 822, respectively. For each vehicle type, we randomly select 200 samples for training the softmax parameters and 200 samples as test samples. In order to give a better estimation of the generalization performance, the reported results of the dataset are the averages of 20 independent experiments.

We also test our method on the dataset in [41]. There are totally 3,618 daylight and 1,306 nightlight images with the image size of 1600 × 1264 in the dataset. All the images are captured in highways with a fixed camera. The vehicles in the images fall into five categories: Truck, Minivan, Bus, Passenger car,

[1]The BIT-Vehicle Dataset can be accessed for research purpose by the link of http://iitlab.bit.edu.cn/mcislab/vehicledb.

Fig. 4. The example images of the dataset in [41]. This dataset consists of 5 types of vehicles: Bus, Minivan, Passenger car, Sedan, and Truck.

TABLE I
CONFUSION MATRIX OF OUR METHOD ON BIT-VEHICLE DATASET

	Bus	Microbus	Minivan	SUV	Sedan	Truck
Bus	0.98	0.00	0.01	0.00	0.00	0.01
Microbus	0.01	0.84	0.05	0.06	0.03	0.01
Minivan	0.01	0.05	0.83	0.00	0.01	0.10
SUV	0.00	0.06	0.01	0.84	0.09	0.00
Sedan	0.00	0.02	0.00	0.06	0.91	0.00
Truck	0.01	0.01	0.09	0.00	0.00	0.90

and Sedan(including sport-utility vehicle (SUV)). The dataset contains some challenging factors, including illumination variations, rain blurring, different color surfaces of vehicles, and background interferences, as shown in Fig. 4. For both datasets, we learn the filters by the SLFL with the parameters as $\alpha = 0.3$, $\beta = 0.2$, and $\gamma = 0.5$. The parameters for solving $\mathcal{W}$ of the softmax classifier are $\lambda = 0.1$, $\mu = 0.4$, and $\eta = 0.1$, and the threshold is set as $\epsilon = 10^{-4}$.

B. Results on BIT-Vehicle Dataset

We test our method on BIT-Vehicle dataset and report the performance. Our approach achieves 88.11% accuracy. The confusion matrix is shown in Table I. From the matrix, we find that most of the misclassifications are between "SUV" and "Sedan". This is because they have quite similar appearances. Fig. 5 shows some of the classified real word images together with their classification results. Our model can precisely classify vehicle types in some challenging situations, such as different lighting conditions, vehicle parts invisible, and viewpoint changes. The primary reason is that our convolutional neural network is able to learn discriminative features for vehicle type classification. As shown in the last row of Fig. 5, most of the misclassifications are due to visually similar appearance patterns in different vehicle types (e.g., "Sedan" and "SUV") and significant image blurring.

To evaluate the effect of filters learned by the sparse Laplacian filter learning (SLFL), we replace them by random values. The classification accuracy is displayed in Table II. It shows that the network with learned filters outperforms that with random filters. The performance is significantly improved by using the

Fig. 5. Some real world images with classification results.

TABLE II
THE CLASSIFICATION ACCURACIES VERSUS DIFFERENT FILTER LEARNING METHODS ON BIT-VEHICLE DATASET

Filter Learning Methods	Accuracy(%)
Randomly	84.22
SLFL without Laplacian	87.18
Sparse Filtering [36]	86.82
SLFL	88.11

SLFL as the sparse filters learned from unlabeled vehicles are able to capture rich discriminative information of vehicles. The effect of the manifold assumption involving in the SLFL is also verified. We simply set β in Eq. (13) as 0 to remove the Laplacian term. The filters are learned and applied in the convolutional neural network. The classification accuracy is also shown in Table II. The performance difference between the "SLFL" and the "SLFL without Laplacian" demonstrates the effectiveness of the manifold assumption during learning filters. The sparse function $\text{sps}(\cdot)$ in Eq. (13) is the same with the objective function of sparse filtering [36]. We use the sparse filtering to learn filters for classification and compare the accuracy with the SLFL. The high performance of "SLFL" shows that the SLFL is more effective in learning filters. The reason is that the SLFL takes the reconstruction, the sparse property, and the manifold assumption all into account, while the sparse filtering method only considers the sparse property.

We further investigate the contribution of the criterion for learning the parameters $\mathcal{W}$ of the softmax layer. We use the simple KL divergence which is the same with the objective function of Eq. (23) as the criterion for learning $\mathcal{W}$. The effectiveness of the constraints for the latent task matrix and the combination coefficient matrix is also evaluated. We remove $\|\mathcal{T}\|_*$ and $\|\mathcal{C}\|_*$ from Eq. (24) and report the classification performances, respectively. The results are all described in Table III. As shown in the table, all the three constraints are beneficial for learning the softmax parameter. The benefits of the observation that many vehicle types are highly correlated can be clearly seen from the accuracy difference between the "KL divergence" and the "Multi-task learning". The learned

TABLE III
CLASSIFICATION ACCURACIES VERSUS DIFFERENT CLASSIFIERS ON BIT-VEHICLE DATASET

Classifiers	Accuracy(%)
KL divergence	86.69
Multi-task Learning without $\|\mathcal{T}\|_*$	85.26
Multi-task Learning without $\|\mathcal{C}\|_*$	86.17
Mult-task Learning	88.11

TABLE IV
CLASSIFICATION ACCURACIES VERSUS DIFFERENT FEATURES ON BIT-VEHICLE DATASET

Features	Accuracy(%)
Only 1st stage feature	85.58
Only 2nd stage feature	87.43
1st + 2nd stage features	88.11

shared knowledge between vehicle types is demonstrated to be effective for classification. The constraints of latent task matrix and coefficient matrix are also beneficial to generate robust model, as illustrated in Table III.

Furthermore, the effect of the network depth and the benefit of the layer-skipping strategy are verified. We evaluate two types of features. For the first one, only the 1st stage is used to learn vehicle features, and the dimensionality of the final feature is 2304. For the second one, the 2nd stage is added but without layer-skipping strategy, and the dimensionality of the final feature is 4096. Their average classification accuracies are shown in Table IV, and the performance difference emphasizes that multi-stage is better than one stage for vehicle feature learning. The layer-skipping strategy connects the features learned from the 1st stage and the 2nd stage. The features learned from the 1st stage are low-level and local, and the outputs of the 2nd stage are high-level global features. Our model uses the high-level global features to capture rich and discriminative information. With low-level local features, our model is able to describe the details of the vehicle precisely. Therefore, these two types of features can be effectively used by adding the layer-skipping strategy into the network, as illustrated in Table IV.

C. Comparison Results

We test our method on the dataset in [41]. Similar to [41], the experiments on daylight images and nightlight images are performed respectively. Since there is only one vehicle in an image of the dataset, the image is directly used as the input of the convolutional neural network to learn features without vehicle detection. The reported results of the dataset are the averages of 20 independent experiments for a better estimation of the generalization performance. Our method achieves 96.1% classification accuracy on daylight images and 89.4% on nightlight images, better than the results of previous methods, as demonstrated in Table V. The underlying reason is that the convolutional neural network we use is able to learn discriminative and reliable features for vehicle type classification. The unsupervised pre-trained filters can capture rich and discriminative information of vehicles. The multi-task learning is able to learn the robust model for classification. In addition, the layer-skipping strategy allows the classifier use both high-level global and low-level local features. It should be noted that our method outperforms other methods even without vehicle detection.

TABLE V
COMPARISON RESULTS OF DIFFERENT METHODS ON THE DATASET IN [41]

Methods	Accuracy(%)	
	Daylight	Nightlight
Psyllos *et al.* [14]	78.3	73.3
Petrovic *et al.* [12]	84.3	82.7
Peng *et al.* [41]	90.0	87.6
Dong *et al.* [42]	91.3	–
Peng *et al.* [16]	93.7	–
Ours	96.1	89.4

V. CONCLUSION

We have proposed a vehicle type classification method from vehicle frontal view images by using a semi-supervised convolutional neural network. The filters of the network are learned by the proposed sparse Laplacian filter learning method to capture rich and discriminative information of vehicles. Serving as the output layer, the softmax classifier is trained by the multi-task learning. The network takes the vehicle image as the input and outputs the probability of each type to which the vehicle belongs. The features learned by the network are discriminative enough to work well in complex scenes. Experimental results on our own BIT-Vehicle dataset and a public dataset demonstrate the effectiveness of the proposed method.

REFERENCES

[1] A. H. Lai, G. S. Fung, and N. H. Yung, "Vehicle type classification from visual-based dimension estimation," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2001, pp. 201–206.
[2] S. Gupte, O. Masoud, R. F. Martin, and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 37–47, Mar. 2002.
[3] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 175–187, Jun. 2006.
[4] Z. Zhang, T. Tan, K. Huang, and Y. Wang, "Three-dimensional deformable-model-based localization and recognition of road vehicles," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 1–13, Jan. 2012.
[5] X. Ma and W. E. L. Grimson, "Edge-based rich representation for vehicle classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, vol. 2, pp. 1185–1192.
[6] C. Zhang, X. Chen, and W.-B. Chen, "A PCA-based vehicle classification framework," in *Proc. IEEE 22nd Int. Conf. Data Eng. Workshops*, 2006, pp. 17–17.
[7] P. Ji, L. Jin, and X. Li, "Vision-based vehicle type classification using partial Gabor filter bank," in *Proc. IEEE Int. Conf. Autom. Logist.*, 2007, pp. 1037–1040.
[8] Y. Shan, H. S. Sawhney, and R. Kumar, "Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 4, pp. 700–711, Apr. 2008.
[9] M. Jiang and H. Li, "Vehicle classification based on hierarchical support vector machine," in *Computer Engineering and Networking*, Dordrecht, Switzerland: Springer-Verlag, 2014, pp. 593–600.
[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Jan. 2004.

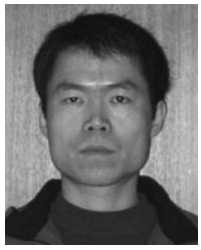
[11] I. Sobel, "Camera models and machine perception," Stanford Univ. Artif. Intell. Lab, Stanford, CA, USA, Tech. Rep. AIM-21, 1970.
[12] V. S. Petrovic and T. F. Cootes, "Analysis of features for rigid structure vehicle type recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2004, pp. 1–10.
[13] P. Negri, X. Clady, M. Milgram, and R. Poulenard, "An oriented-contour point based voting algorithm for vehicle type classification," in *Proc. IEEE Int. Conf. Pattern Recog.*, 2006, vol. 1, pp. 574–577.
[14] A. Psyllos, C.-N. Anagnostopoulos, and E. Kayafas, "Vehicle model recognition from frontal view image measurements," *Comput. Std. Interfaces*, vol. 33, no. 2, pp. 142–151, Jun. 2011.
[15] B. Zhang, "Reliable classification of vehicle types based on cascade classifier ensembles," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 1, pp. 322–332, Mar. 2013.
[16] Y. Peng *et al.*, "Vehicle type classification using data mining techniques," in *The Era of Interactive Media*, New York, NY, USA: Springer-Verlag, 2013, pp. 325–335.
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[18] C. Garcia and M. Delakis, "A neural architecture for fast and robust face detection," in *Proc. IEEE 16th Int. Conf. Pattern Recog.*, 2002, vol. 2, pp. 44–47.
[19] M. Osadchy, Y. L. Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *J. Mach. Learn. Res.*, vol. 8, pp. 1197–1215, 2007.
[20] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 3476–3483.
[21] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE CVPR*, 2013, pp. 3626–3633.
[22] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, "PANDA: Pose aligned networks for deep attribute modeling," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1637–1644.
[23] L. Kang, P. Ye, Y. Li, and D. Doermann, "Convolutional neural networks for no-reference image quality assessment," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1733–1740.
[24] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.
[25] A. Karpathy *et al.*, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1725–1732.
[26] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[27] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, vol. 19, pp. 153–160.
[28] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 2146–2153.
[29] K. Kavukcuoglu *et al.*, "Learning convolutional feature hierarchies for visual recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1090–1098.
[30] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.
[31] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1383–1390.
[32] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962.
[33] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognit.*, vol. 15, no. 6, pp. 455–469, 1982.
[34] S. Lyu and E. P. Simoncelli, "Nonlinear image representation using divisive normalization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2008, pp. 1–8.
[35] N. Pinto, D. D. Cox, and J. J. DiCarlo, "Why is real-world visual object recognition hard?" *PLoS Comput. Biol.*, vol. 4, no. 1, p. e27, Jan. 2008.
[36] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Ng, "Sparse filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1125–1133.
[37] D. J. Field, "What is the goal of sensory coding?" *Neural Comput.*, vol. 6, no. 4, pp. 559–601, Jul. 1994.
[38] B. Willmore and D. J. Tolhurst, "Characterizing the sparseness of neural codes," *Netw., Comput. Neural Syst.*, vol. 12, no. 3, pp. 255–270, Aug. 2001.
[39] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.*, vol. 35, no. 151, pp. 773–782, Jul. 1980.
[40] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.
[41] Y. Peng, J. S. Jin, S. Luo, M. Xu, and Y. Cui, "Vehicle type classification using PCA with self-clustering," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2012, pp. 384–389.
[42] D. Zhen and Y. Jia, "Vehicle type classification using distributions of structural and appearance-based features," in *Proc. Int. Conf. Image Process.*, 2013, pp. 4321–4324.

Zhen Dong received the B.S. degree in computer science in 2011 from Beijing Institute of Technology, Beijing, China, where he is currently a Ph.D. candidate. His research interests include computer vision and machine learning.

Yuwei Wu received the Ph.D. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2014.

He is a Research Fellow with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests are in computer vision, video analytics, machine learning, and pattern recognition.

Dr. Wu received the National Scholarship for Graduate Students and the Academic Scholarship for Ph.D. Candidates from the Ministry of Education of China, the Outstanding Ph.D. Thesis Award and the Xu Teli Excellent Scholarship from BIT, and the CASC Scholarship from China Aerospace Science and Industry Corporation (CASIC).



Mingtao Pei received the Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004.

He is an Associate Professor with the School of Computer Science, Beijing Institute of Technology. From 2009 to 2011, he was a Visiting Scholar with the Center for Image and Vision Science, University of California, Los Angeles. His main research interest is computer vision with an emphasis on event recognition and machine learning.

Dr. Pei is a member of China Computer Federation.

Yunde Jia (M'11) is a Professor of computer science with Beijing Institute of Technology, Beijing, China.

He is the Director of Beijing Laboratory of Intelligent Information Technology. He was a Visiting Scientist at Carnegie Mellon University between 1995 and 1997 and a Visiting Fellow with the Australian National University in 2011. His research interests include computer vision, media computing, and intelligent systems.