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.)
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 $l_0$, $l_1$ or $l_2$ norms, the SLFL uses the sparse function $sps(\cdot)$ 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 $l_1$-norm. The $l_1$-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 filter 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 non-linear 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 $x$ and $y$ to represent the input and output of each layer, respectively. They are both 3D arrays where $x$ are with the size of $s_1 \times s_2 \times s_3$ and $y$ with the size of $t_1 \times t_2 \times t_3$.

A. Convolutional Layer

In the convolutional layer, convolutions between the input and a series of filters are first computed. An element-wise non-linear activation function is then executed on the convolutions.
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 \( x \) is a 3D array with the size of \( s_1 \times s_2 \times s_3 \), where \( s_3 \) is the number of 2D feature maps, and \( s_1 \times s_2 \) is the size of the 2D feature map which is represented by \( x_i \). The output \( y \) is also a 3D array whose size is \( t_1 \times t_2 \times t_3 \). Similar to the 2D feature map \( x_i \) of the input, \( y_j \) is defined as the \( j \)-th 2D feature map of the output. The element-wise sigmoid function \( \text{sig}(\cdot) \) is chosen as the non-linear activation function. Hence, \( y_j \) is computed by

\[
y_j = \text{sig} \left( \sum_i k_{ij} \otimes x_i \right),
\]

where \( \otimes \) denotes the convolution operation, and \( k_{ij} \) is a 2D filter learned by the sparse Laplacian filter learning described in Section III-A. Suppose that the filter size is \( l_1 \times l_2 \), we have \( t_1 = s_1 - l_1 + 1 \) and \( t_2 = s_2 - l_2 + 1 \) 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 \( 143 \times 143 \) to take a balance between the computation time cost and the accuracy. The size of output 2D feature map is \( 135 \times 135 \) since the filters are with the size of \( 9 \times 9 \) 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

\[
y_{i,j,k} = |x_{i,j,k}|,
\]

where \( x_{i,j,k} \) and \( y_{i,j,k} \) represent each element of \( x \) and \( y \), 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 \( x_{i,j,k} \) in the input 3D array size of \( s_1 \times s_2 \times s_3 \), the subtractive normalization operator is given by

\[
z_{i,j,k} = x_{i,j,k} - \sum_{p=-4}^{4} \sum_{q=-4}^{4} \sum_{r=-1}^{s_3} \omega_{p,q} x_{i+p,j+q,r},
\]

where \( \omega_{p,q} \) is a normalized Gaussian filter with the size of \( 9 \times 9 \). \( z \) is the output of the subtractive normalization and the input of the divisive normalization. The divisive normalization operator is defined as

\[
y_{i,j,k} = \frac{z_{i,j,k}}{\max(M(i,j))},
\]

where

\[
M(i,j) = \sqrt{\sum_{p=-4}^{4} \sum_{q=-4}^{4} \sum_{r=-1}^{s_3} \omega_{p,q} z_{i+p,j+q,r}^2},
\]

\[
M = \left( \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} M(i,j) \right) / (s_1 \times s_2).
\]
In both operations, the filtering by \( \omega_{p,q} \) 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

\[
y_{i,j,k} = \sum_{p,q} \alpha_{p,q} x_{i+p,j+q,k},
\]

where \( \alpha_{p,q} = 1/(f_1 \times f_2) \) is the average filter with the size of \( f_1 \times f_2 \), \( x \) and \( y \) 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 \( p_1 \) horizontally and \( p_2 \) vertically. Suppose that the input 2D feature maps of these two layers are size of \( s_1 \times s_2 \), and the size of the output 2D feature map is \( t_1 \times t_2 \), we have

\[
t_1 = \left\lfloor \frac{s_1 - f_1}{p_1} \right\rfloor + 1,
\]

\[
t_2 = \left\lfloor \frac{s_2 - f_2}{p_2} \right\rfloor + 1,
\]

where \( \lceil \cdot \rceil \) denotes the maximum integer less or equal than \( \delta \). For example, the input 3D array of the pooling layer in the 1st stage is with the width and height of \( s_1 = s_2 = 135 \). The appropriate average filter size and subsampling rate are set for simplicity. The average filter size is \( f_1 = f_2 = 10 \), and the subsampling rates are 5 in both the horizontal direction and the vertical direction. Therefore, the size of the output 2D array is \( t_1 = t_2 = \left\lfloor (135 - 10)/5 \right\rfloor + 1 = 26 \).

### 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

\[
v = W^\top x + b,
\]

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

\[
v = W^\top x + b = \begin{bmatrix} W^\top b \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = W^\top \begin{bmatrix} x \end{bmatrix},
\]

where \( W = [w_1, w_2, \ldots, w_C] \in \mathbb{R}^{(D+1) \times C} \), and each column of \( W \) is the corresponding vehicle type model parameter. Because the probability has the properties of nonnegativity and unitarity, \( v \) is normalized as

\[
y_i = \frac{1}{\sqrt{v_i}}, \quad i = 1, 2, \ldots, C,
\]

\[
V = \sum_{i=1}^{C} e^{v_i},
\]

where \( v_i \) is the \( i \)-th element of \( v \), and \( y = [y_1, y_2, \ldots, y_C]^\top \) 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 \( U = [u_1, u_2, \ldots, u_n] \in \mathbb{R}^{d \times n} \), where the columns are data points. Our goal is to learn the filter bank \( K \in \mathbb{R}^{d \times t} \) which consists of \( t \) filters. Using this filter bank, the input data points \( U \) can be mapped to sparse representations \( A \). The nonlinear map function is given by

\[
A = \text{sig}(K^\top U),
\]

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

\[
\min_{B, A, K} \|U - BA\|_F^2 + \alpha \text{sp}(A)
\]

\[
+ \beta \text{tr}(ACAC^\top) + \gamma \|A - \text{sig}(K^\top U)\|_F^2,
\]

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

The \( \text{sp}(\cdot) \) 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 \( A_{(i,\cdot)} \in \mathbb{R}^{1 \times t}(i = 1, 2, \ldots, t) \) be the \( i \)-th row of \( A \), and \( A_{(\cdot,j)} \in \mathbb{R}^{t \times 1}(j = 1, 2, \ldots, n) \) the \( j \)-th column of \( A \). The \( \text{sp}(\cdot) \) 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., $A_{(i,j)} = A_{(i,j)} / \|A_{(i,j)}\|_2$. In this way, the feature is equally active. In the second step, each example is divided by its $\ell_2$-norm across all features, $\hat{A}_{(i,j)} = \hat{A}_{(i,j)} / \|\hat{A}_{(i,j)}\|_2$, 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 $\hat{A}$. The $\text{sp}(\cdot)$ function is given by

$$\min_K \|\hat{A}\|_* = \sum_{i=1}^n \sum_{j=1}^n |\hat{A}_{i,j}|$$

$$= \sum_{j=1}^n \|\hat{A}_{(i,j)}\|_1 = \sum_{j=1}^n \frac{\|\hat{A}_{(i,j)}\|_2}{\|\hat{A}_{(i,j)}\|_2}, \quad (14)$$

where $\|M\|_*$ denotes summing up the absolute values of all the elements in the matrix $M$. The minimization of the sparse function makes the feature distribution $A$ 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 $\|\hat{A}_{(i,j)}\|_1 = \|\hat{A}_{(i,j)}\|_2 / \|\hat{A}_{(i,j)}\|_2$ in Eq.(14) reflects the population sparsity property of the features on the $j$-th example. Since $\hat{A}_{(i,j)}$ 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

$$T_i = \frac{1}{n} \sum_{j=1}^n A_{ij}^2 = \frac{1}{n} \|A_{(i,:)}\|_2^2. \quad (15)$$

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 $x_i$, and the edge weight matrix $R$ is defined as

$$R_{ij} = \begin{cases} \frac{u_i^T u_j}{\|u_i\| \|u_j\|} & \text{if } u_i \in N_{\epsilon}(u_j) \text{ or } u_j \in N_{\epsilon}(u_i) \\ 0 & \text{otherwise,} \end{cases} \quad (16)$$

where $N_{\epsilon}(u_i)$ represents the set of $\epsilon$ nearest neighbors of $u_i$. The edge weight matrix satisfies that large values $R_{ij}$ is corresponding to nearby data points. The manifold assumption is formulated as the minimization of

$$\frac{1}{2} \sum_{i,j=1}^n \|A_{(i,j)} - A_{(i,j)}\|^2 = \text{Tr}(A^T D A), \quad (17)$$

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

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

**STEP 1:** Keeping parameters $\beta$ and $K$ fixed, learn the representation $A$ by solving the optimization problem:

$$\min_A \|U - BA\|_F^2 + \alpha \text{sp}(\cdot) \quad \text{subject to } A \geq 0.$$  \quad (18)

In our implementation, the L-BFGS optimization method [39] is used. The gradient of the objective function in Eq. (18) with respect to $A$ is easy to solve. The gradient of $\text{sp}(\cdot)$ is computed by the back propagation algorithm. As the $\text{sp}(\cdot)$ 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 $A$ from STEP 1, minimize Eq. (13) with respect to $\beta$ and $K$. The optimization problem is rewritten as

$$\min_{\beta, K} \|U - BA\|_F^2 + \gamma \|A - \text{sig}(K^T U)\|_F^2. \quad (19)$$

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

$$\min_B \|U - BA\|_F^2. \quad (20)$$

An analytical solution of $B$ is obtained that $B = U^T (A^T A)^{-1}$. The columns of $B$ 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, $K$ 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 $\mathcal{K}$:

$$
\min_{\mathcal{K}} \| A - \text{sig}(\mathcal{K}^T U) \|^2_F. \tag{21}
$$

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**

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

1. **Step 1. Initialization**
   1. Initialize dictionary $B$ by using k-means method on $U$;
   2. Normalize each column of $B$ to unit norm;
   3. $A = (B^T B)^{−1} B^T U$;
   4. $K = (U A^T)^{−1} U A^T$;

2. **Step 2. Optimize $A$ with fixed $B$ and $K$**
   1. Solve Eq.(18) to obtain $A$ by L-BFGS;

3. **Step 3. Keeping $A$ fixed, optimize $B$ and $K$**
   1. $B = U A^T (AA^T)^{−1}$;
   2. Normalize each column of $B$ to unit norm;
   3. Solve Eq.(21) to obtain $K$ by L-BFGS;

4. **Step 4. Output $B$, $A$ and $K$.**

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 $\mathcal{T} \in \mathbb{R}^{(t+1) \times K}$ 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 $C \in \mathbb{R}^{K \times C}$ with each column representing the combination coefficients of the corresponding vehicle type. We thus have

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

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

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

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

where $d^{(i)}_j$ and $y^{(i)}_j$ represent the $j$-th element of $d^{(i)}$ and $y^{(i)}$, 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 over-fitting. Taking these three constraints into account, the objective function for learning $\mathcal{T}$ and $\mathcal{C}$ is formulated as

$$
\begin{align*}
\min_{\mathcal{T}, \mathcal{C}} & - \sum_{i=1}^{N} \sum_{j=1}^{C} d^{(i)}_j \ln y^{(i)}_j + \lambda \| \mathcal{T} \|^2_F \\
& + \mu \| \mathcal{T} \|_1 + \eta \| \mathcal{C} \|_\ast, \tag{24}
\end{align*}
$$

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

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

The objective function is not smooth with respect to $\mathcal{C}$ as the existence of $\| \cdot \|_\ast$. 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

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

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

$$C_i = \frac{r_{i-1}^2 + r_{i-2}^2 - 1}{r_{i-1}^2} C_{i-1} - \frac{r_{i-2}^2 - 1}{r_{i-1}^2} C_{i-2},$$

$C_i = h \left( C_i - \frac{1}{\xi} \nabla f(C_i) ; \eta \right), \quad (27)$

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

**STEP 2:** Keeping $C$ fixed, the optimal latent task matrix $T$ is learned by solving

$$\min_T - \sum_{i=1}^{N} \sum_{j=1}^{C} d_{ij}^{(i)} \ln y_{ij}^{(i)} + \lambda \|T\|_F^2 + \mu \|T\|_* . \quad (28)$$

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

$$f(T) = - \sum_{i=1}^{N} \sum_{j=1}^{C} d_{ij}^{(i)} \ln y_{ij}^{(i)} + \lambda \|T\|_F^2 ,$$

$$g(T) = \mu \|T\|_* . \quad (29)$$

The overall algorithm for learning $W$ 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 $O(\sqrt{k_1})$ 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.

**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 $1600 \times 1200$ and $1920 \times 1080$ 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 are divided into six categories: Bus, Microbus, Minivan, Sedan, SUV, and Truck.

1The 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

<table>
<thead>
<tr>
<th></th>
<th>Bus</th>
<th>Microbus</th>
<th>Minivan</th>
<th>SUV</th>
<th>Sedan</th>
<th>Truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus</td>
<td>0.98</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>Microbus</td>
<td>0.01</td>
<td>0.84</td>
<td>0.05</td>
<td>0.06</td>
<td>0.03</td>
<td>0.01</td>
</tr>
<tr>
<td>Minivan</td>
<td>0.01</td>
<td>0.05</td>
<td>0.83</td>
<td>0.00</td>
<td>0.01</td>
<td>0.10</td>
</tr>
<tr>
<td>SUV</td>
<td>0.00</td>
<td>0.06</td>
<td>0.01</td>
<td>0.84</td>
<td>0.09</td>
<td>0.00</td>
</tr>
<tr>
<td>Sedan</td>
<td>0.00</td>
<td>0.02</td>
<td>0.00</td>
<td>0.06</td>
<td>0.91</td>
<td>0.00</td>
</tr>
<tr>
<td>Truck</td>
<td>0.01</td>
<td>0.01</td>
<td>0.09</td>
<td>0.00</td>
<td>0.00</td>
<td>0.90</td>
</tr>
</tbody>
</table>

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 \( W \) 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. As shown in the table, the network with learned filters outperforms that with random filters. The performance is significantly improved by using the 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 \( \beta \) 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 \( W \) 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 \( W \). The effectiveness of the constraints for the latent task matrix and the combination coefficient matrix is also evaluated. We remove \( \| T \|_* \) and \( |C|_* \) 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
unsupervised pre-trained filters can capture rich and discrimi-
native and reliable features for vehicle type classification. The
convolutional neural network we use is able to learn discrimi-
nation of the generalization performance. Our method achieves
averages of 20 independent experiments for a better estima-
tive 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.

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 discrimin-
itive and reliable features for vehicle type classification. The
unsupervised pre-trained filters can capture rich and discrimina-
tive features 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.

V. Conclusion

We have proposed a vehicle type classification method from
vehicle frontal view images by using a semi-supervised convo-
lutional 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 demon-
strate the effectiveness of the proposed method.

 REFERENCES

surveillance system for vehicle tracking and classification,” *IEEE Trans.
deformable-model-based localization and recognition of road vehicles,”
pp. 1185–1192.
framework,” in *Proc. IEEE 22nd Int. Conf. Data Eng. Workshops*, 2006,
pp. 17–17.
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 dis-
criminative edge measures for vehicle matching between nonoverlap-
vector machine,” in *Computer Engineering and Networking*, Dordrecht,
Switzerland: Springer-Verlag, 2014, pp. 593–600.


