Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks

Xueyun Chen, Shiming Xiang, Cheng-Lin Liu, and Chun-Hong Pan

***Abstract*—Detecting small objects such as vehicles in satellite images is a difficult problem. Many features (such as histogram of oriented gradient, local binary pattern, scale-invariant feature transform, etc.) have been used to improve the performance of object detection, but mostly in simple environments such as those on roads. Kembhavi *et al.* proposed that no satisfactory accuracy has been achieved in complex environments such as the City of San Francisco. Deep convolutional neural networks (DNNs) can learn rich features from the training data automatically and has achieved state-of-the-art performance in many image classification databases. Though the DNN has shown robustness to distortion, it only extracts features of the same scale, and hence is insufficient to tolerate large-scale variance of object. In this letter, we present a hybrid DNN (HDNN), by dividing the maps of the last convolutional layer and the max-pooling layer of DNN into multiple blocks of variable receptive field sizes or max-pooling field sizes, to enable the HDNN to extract variable-scale features. Comparative experimental results indicate that our proposed HDNN significantly outperforms the traditional DNN on vehicle detection.**

***Index Terms*—Deep convolutional neural networks (DNNs), hybrid DNNs (HDNNs), remote sensing, vehicle detection.**

I. Introduction

DETECTING vehicles in high-resolution satellite images has wide applications in military and homeland surveillance, transportation planning, and intelligent traffic guidance systems. Many works have been done [1]–[7], but it still remains a highly challenging task.

The performance of object detection relies on the features extracted from the object. Zhao and Nevatia [1] combined the geometric boundary of the car body, the front windshield, and the shadow into a Bayesian network. Eikvil *et al.* [2] combined geometric-shape properties, gray level features, and Hu moments to detect vehicles in high-resolution satellite images. Dalal and Triggs [15] proposed a method for object detection using histogram of oriented gradients (HOGs) and linear support vector machine (SVM) for classification. They revealed the preferability of gradient-based features and reported superior object detection performance. Later, many vehicle detection methods using variations of HOG, and different classifiers were proposed.

Manuscript received August 30, 2013; revised December 10, 2013 and January 16, 2014; accepted January 30, 2014. Date of publication March 25, 2014; date of current version May 12, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 91338202, by the National Basic Research Program of China (973 Program) under Grant 2012CB316300, and by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06030300.

The authors are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: xueyun.chen@nlpr.ia.ac.cn; smxiang@nlpr.ia.ac.cn; liucl@nlpr.ia.ac.cn; chpan@nlpr.ia.ac.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/LGRS.2014.2309695

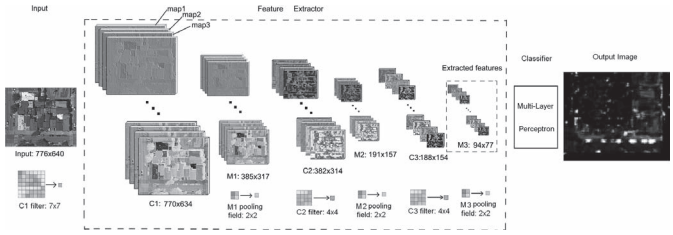

Fig. 1. Example of DNN. Where $n_l = 3$, the maps in M^3 have size 94×77, and the feature scale of each pixel is 28×28.

Liang *et al.* [3] combined HOG and Haar wave features with a multiple kernel SVM (MKL-SVM). MKL-SVM is an extension to the standard SVM, in which multiple kernels with different parameters are trained and integrated by linear combination. However, MKL-SVM is not suitable for large-scale data because it needs much more training time than SVM. Grabner *et al.* [4] integrated HOG, Haar wavelets, and local binary patterns (LBPs) [16] using an Adaboost classifier. Sun *et al.* [8] also combined multiple features using Adaboost. Kembhavi *et al.* [5] extracted multiscale HOG features from the original image, six-color probability maps, and two pairs of pixel maps. They reduced the feature dimensionality by a partial least square. Ali *et al.* [6] designed a pose-indexed feature using an Adaboost classifier. More features used in general object detection include the keypoints [9] and the embedded taxonomic semantics of object categories [10].

All the above methods are based on handcrafted features, which cannot reach an optimal balance between the discriminability and the robustness without considering the details of real data. An alternative way is to learn rich features from the training set automatically. The deep convolutional neural network (DNN) [10], as a feature learning architecture, has yielded superior performance in many object recognition tasks. The DNN uses the convolution layers and the max-pooling layers. The hidden layers and the output layer combine the extracted features for classification.

The layers of DNN can be divided into two parts. One is the feature extractor, which extracts features hierarchically using the convolutional layers and the max-pooling layers. The other is a multilayer perceptron (MLP) classifier, which classifies the data by the extracted features. The MLP is composed of the hidden layers and the output layer.

Each convolutional layer of DNN generates feature maps using sliding filters (templates) on a local receptive field in the maps of the preceding layer (input or max-pooling layer). The map sizes decrease layer by layer such that the extracted feature becomes more complex and global.

Compared with the traditional convolutional neural networks (CNNs) (only a few maps, no more than 6 layers), the DNN is

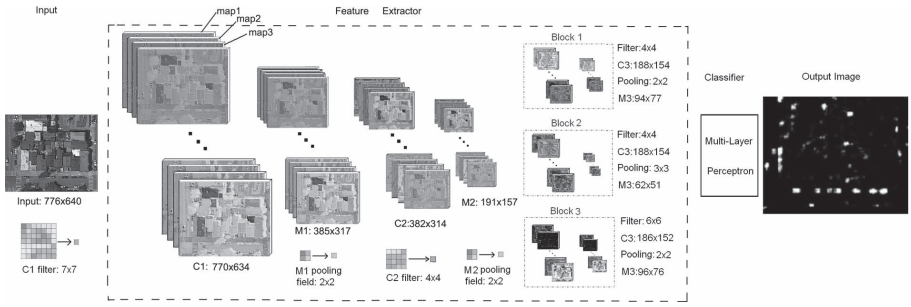

Fig. 2. Example of HDNN, where $n_l = 3$, $n_b = 3$. The M^3 map sizes of B_1, B_2, and B_3 are 94×77, 62×51, and 96×76, respectively. Their pixel feature scales are 28×28, 28×28, and 36×36, respectively.

more deep (6–10 layers), and wider (40–400 maps). The more templates (filters) contained by DNN, the more target's shape variants can be detected by it.

The concept "receptive field" (the filter definition field) originates from Hubel and Wiesel's study [11] on cats' striate cortex. The max-pooling layer was first introduced by Fukushima [12]. The normal structure of CNN was proposed by LeCun *et al.* [13] who first used the concept "convolutional layer." Recently, Ciresan *et al.* [10] have presented the structure of DNNs and achieved the state-of-the-art performance on six benchmark image classification databases, including the MNIST (handwritten digits), NIST SD-19, handwritten Chinese characters, traffic signs, CIFAR10, and NORB. The results in MNIST and traffic signs are even superior to human performance.

The CNN has been used successfully in object detection for a long time. Garcia and Delakis [14] used a six-layer CNN for face detection in CMU and MIT test sets. Despite the demonstrated superiority of the DNN, its every convolutional layer extracts features of the same scale only. This limits its performance in the case of variable-scale objects. Even for objects of the same scale, multiscale features are beneficial for classification because they may improve the separability of the feature space. We thus present a hybrid DNN (HDNN) for vehicle detection. HDNN is capable to extract multiscale features at its highest convolutional layer. Our experimental results in vehicle detection justify the benefits of this multiscale structure.

II. Architecture of HDNN

Here, we first discuss the structure of DNN. We give a strict definition of the concept "feature scale," deduce a proposition that DNN only extracts features of the same scale. Then, we show that the blocks of HDNN can extract multiscale features. Finally, we explain why the blocks of HDNN only include the highest convolutional and max-pooling layers.

A. Structure of DNN

The layers of DNN can be divided into two parts, i.e., feature extractor and MLP classifier. Let n_l denote the number of the convolutional layers, and n_m denotes the number of maps in a convolutional layer. For convenience, we suppose all convolutional layers have the same number of maps. The convolutional layers of DNN are denoted by $C^1, \ldots, C^{n_l}$, and the max-pooling layers of DNN are denoted by $M^1, \ldots, M^{n_l}$. All the convolutional and max-pooling layers compose the feature extractor of DNN. The layer M^{n_l} outputs the extracted features to the MLP Classifier.

The MLP includes the hidden layers and the output layer. Its output value can be transformed into the output image (right part of Fig. 1), where the brightness of pixels signifies the likelihood of vehicle detection. The $\tanh$ function is used as the kernel function for all the nodes in DNN.

Fig. 1 gives an example of DNN. The convolutional layer maps are determined by the filters sliding on the previous layer pixel by pixel. The max-pooling layer maps are determined by the max-pooling function on the nonoverlapped max-pooling fields sliding over the previous convolutional layer. The max-pooling function is to reduce the map size, enhance the shift-invariant ability, and antinoise ability by the "winner-take-all" principle.

Definition 1: The **source area** of a node (pixel) of DNN includes all its relevant pixels in the input image.

Definition 2: The **feature scale** of a node (pixel) of DNN is the largest size of its possible source areas.

In Fig. 1, all pixels of M^3 have passed three filters: 7×7 in C^1, 4×4 in C^2, and 4×4 in C^3. Simple computing $((4 \times 2 + 3) \times 2 + 6 = 28)$ shows its largest source area size at 28×28. Because it also has passed three max-pooling functions, we do not know the exact source area size (which varies from 22×22 to 28×28).

Proposition 1: DNN only extracts features of the same scale.

B. Structure of HDNN

Different from DNN, the set $\{C^{n_l}, M^{n_l}\}$ of HDNN is divided into n_b blocks: $\{B_1, \ldots, B_{n_b}\}$. Each block has a different filter size or max-pooling field size.

Fig. 2 gives examples of HDNN, the set $\{C^3, M^3\}$ is divided into three blocks: $\{B_1, B_2, B_3\}$. The pixels of M^3 in B_1 and B_2 have passed three filters: 7×7 in C^1, 4×4 in C^2, and 4×4 in C^3, their feature scales are 28×28. The pixels of B_3 have a different filter size: 6×6 in C^3, simple computing $((6 \times 2 + 3) \times 2 + 6 = 36)$ shows that the B_3 feature scales are: 36×36. Thus, this HDNN extracts features of two scales, i.e., 28×28 and 36×36.

In the case when $n_b = 3$, we define HDNN $(n_1 - n_2 - n_3)$ as the HDNN where B_1 has n_1 maps, B_2 has n_2 maps, and B_3 has n_3 maps. The details of training HDNN are provided in the supplemental material.

C. Why the Blocks of HDNN Only Include the Highest Convolutional and Max-Pooling Layers

Suppose $C_i^l(x, y)$ is the pixel (x, y) value of the ith map of C^l, M_j^{l-1} denotes the jth map of M^{l-1}. Let $filter_{ij}^l$ denote the

filter connecting C_i^l to M_j^{l-1}. All filters of C_i^l have the same size, i.e., $L \times H$. b_i^l is the bias of the C_i^l. For both DNN and HDNN, we have

$$C_i^l(x,y) = \tanh\left(\sum_{j=1}^{n_m}\left(\int_0^{L-1}\int_0^{H-1} filter_{ij}^l(u,v) \times M_j^{l-1}(u+x,u+y)dudv\right) + b_i^l\right) \quad (1)$$

where $l = 2,\ldots,n_l,\ i = 1,\ldots,n_m$. Formula (1) reveals all maps of M^{l-1} should have the same size. Otherwise, the summing of all filter convolutional results will not be possible.

From the same map size of M_j^{l-1}, we can deduce that all C_j^{l-1} have the same map size, and so on. This means that all the maps of C^{l-1} have the same map size and filter size, all the maps of M^{l-1} have the same map size and max-pooling field size, i.e., $l = 2,\ldots,n_l$. This is why the blocks of HDNN only include C^{n_l} and M^{n_l}.

There are some other ways to introduce multiscale features into DNN. One way is to supply the results collected from two or more max-pooling layers directly into the MLP classifier. Another way is to neglect the full connections between layer C^l and layer M^{l-1}. By contrast, the structure of the DNN constructed in the former way is very simple. However, such a DNN may contain nonsymmetrical connections between nodes. As a result, the nonsymmetry in this DNN will increase the computation burden. When the latter reduces the total number of filters, it probably diminishes the performance of DNN.

III. Implementation Detail

Here, we first discuss the multiple gradients used in the object's edge extraction, then, we explain the sliding window technique used in vehicle location. Finally, we illustrate the structure of the HDNN used in our experiments.

A. Multiple Gradient Images

We rely on edge information to locate a vehicle. Dalal and Triggs [15] presented a gradient computing method based on the maximal gradient norm of three RGB channels. However, for cars with black color and cars sheltered by the trees, their method only generates dim edges of the black cars and fails to separate the sheltered cars from the textures of the trees. We solve this problem by computing multiple gradients on multiple thresholding images. In Fig. 3(b), the black cars are too dim, and some cars are sheltered by the trees. In Fig. 3(c), the borders of the dim black cars are enhanced. In Fig. 3(d), the tree textures are erased. The two thresholds 60 and 100 are determined by experiments. In theory, more thresholding images help to increase the locating accuracy but generate more repetitive sliding windows at all. In the experiments of this letter, two thresholds are enough. For other kinds of targets such as the aircrafts, three or more thresholding images are recommended [17].

B. Sliding Window Technique

While the usual sliding window technique slides the windows at a fix sliding step, it cannot ensure that the windows cover the vehicles exactly. To this end, we slide each window to its geometric center (computed by its contained image content). However, if the initial window only covers a minor part of the vehicle, sliding once is not enough. Thus, we enlarge the window to contain the main part of the vehicle and slide the window again to its new geometric center.

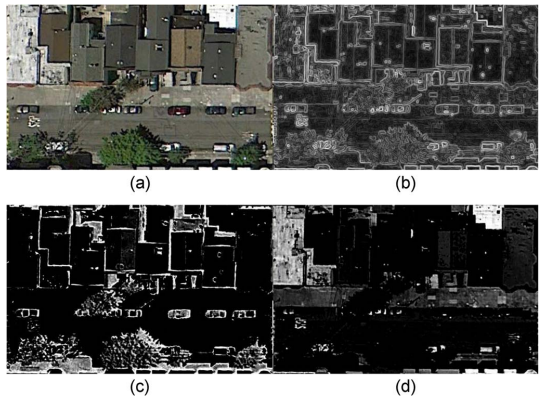

Fig. 3. (a) Original image. (b) Gradient of Dalal and Triggs [15]. (c) Gradient on image thresholding below 60. (d) Gradient on image thresholding from 100.

Algorithm 1 *Vehicle Locating*

Input: Three gradient images, sliding window scale, sliding step, enlarging factor, minimal distance limit.
Output: Vehicle locating windows.
1: On each gradient image, generate the sliding window grid to cover the whole image.
2: For each sliding window W_p at position $p = (x_0, y_0)$, compute its geometric center $p1 = (x_1, y_1)$, center W_p on $p1$ to get a new window W_{p1}.
3: Enlarge the window scale of W_{p1} by the enlarging factor, compute the new geometric center $p2$. Center W_{p1} on $p2$.
4: Collect all sliding windows of the three gradient images, filter the repetitive windows by the minimal distance limit, send the remained windows to the HDNN classifier for vehicle detection.

In algorithm 1, the sliding step is 0.5 $\times$ window scale, and the enlarging factor is 1.414. The minimal distance limit is 0.15 $\times$ window scale. Such parameter values are determined by experiments. For other shape targets, they can be adjusted to get the best results.

Fig. 4(a)–(e) shows the process of algorithm 1 clearly. Fig. 4(f) shows the locating results on a large park where vehicles of variable colors arrange densely. The locating accuracy is based on the following definitions:

Definition 3: A locating window is exact only if the distance between the window center and a vehicle center is less than 0.45 $\times$ window scale.

Definition 4: A vehicle is exactly located if it has at least one exact locating window.

All locating windows are normalized into 48 $\times$ 48 size, and their gray scales are normalized into [0, 255] range. At last, we send them to the HDNN classifier for vehicle detection. In our vehicle database, 99.7% vehicles are located exactly.

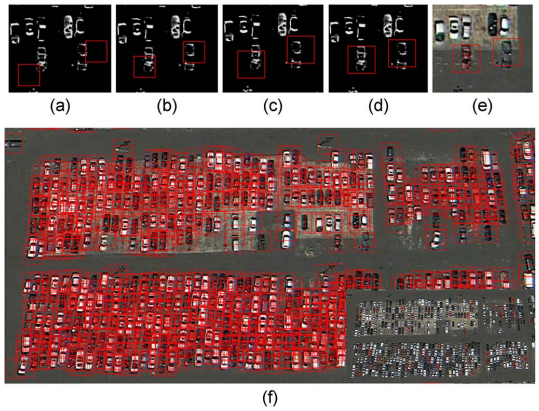

Fig. 4. (a)–(e) Process of locating vehicle by the algorithm 1 on the negative gradient image (threshold = 60). (f) Final locating window set, 99.2% vehicle are exactly located in this complex large park. (a) Initial windows. (b) Move to center. (c) Enlarge twice. (d) Move to center. (e) Original image. (f) The locating window set after filtering repetitive sliding windows.

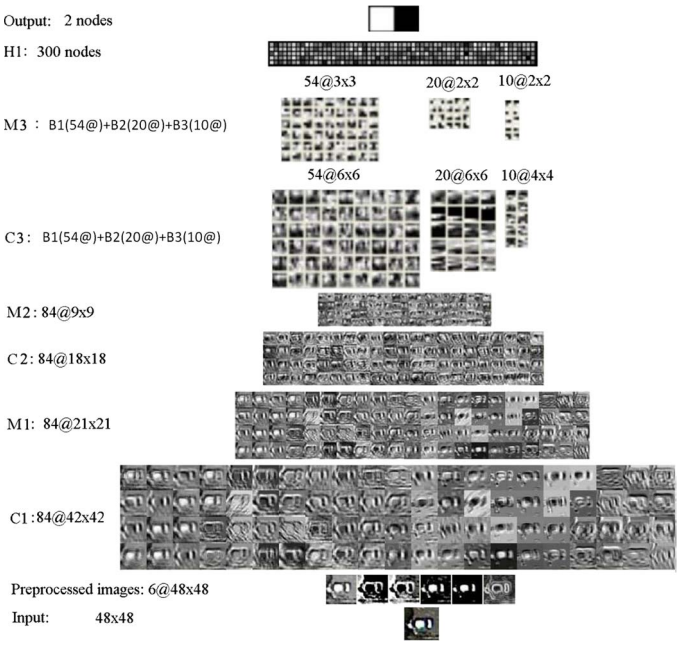

Fig. 5. Structure of HDNN (54-20-10). Where $n_l = 3$, $n_m = 84$, and $n_b = 3$. B_1 has 54 maps, B_2 has 20 maps, and B_3 has 10 maps.

C. Structure of Our HDNN

Fig. 5 shows the structure of the HDNN (54-20-10) we used in vehicle detection. Here, $n_l = 3$, $n_m = 84$, $n_b = 3$. There is only one hidden layer H_1 which has 300 nodes, and the output layer has 2 nodes. B_1 has 54 maps, B_2 has 20 maps, and B_3 has 10 maps.

From the 48 × 48 gray image, we get six preprocessed images, including the gray image, the negative thresholding images at 60 and 100, the thresholding images at 100 and 160, and the gradient image. Each preprocessed image is connected with 14 maps of C^1.

The convolutional filter size of C^1 is 7 × 7, that of C^2 is 4 × 4, that of B_1 is 4 × 4, that of B_2 is 4 × 4, and that of B_3 is 6 × 6. The max-pooling field sizes of M^1 and M^2 are 2 × 2, that of B_1 and B_3 are 2 × 2, and that of B_2 is 3 × 3. The feature scales of B_1 and B_2 are 28 × 28, and the feature scale of B_3 is 36 × 36. The max-pooling map size of B_1 are 3 × 3, that of B_2 are 2 × 2, and that of B_3 are 2 × 2.

We trained HDNN by the back-propagation algorithm on the graphics processing unit (GPU) card, initial weights were set by an uniform random distribution in the range [−0.05, 0.05], all initial biases were set to zero. LearnRate = 0.001, Momentum = 0, WeightDecay = 0, batch size = 50. While training is ended when the validation error is near-zero, it usually needs 5–6 days on our GPU cards. Testing an image in GPU needs about 7–8 s.

TABLE I
FAR OF DIFFERENT METHODS ON VEHICLE TEST SET

Method	train	Given Recall Rate					
		95%	90%	85%	80%	75%	70%
DNN	5.81	23.5	12.2	7.50	5.07	3.45	2.61
HOG+SVM [15]	3.25	67.5	43.4	29.3	20.2	14.3	10.3
LBP+SVM [16]	7.84	87.6	59.2	43.0	32.8	24.5	19.4
Adaboost [18]	2.31	91.6	65.3	49.1	40.1	31.6	25.8

TABLE II
FAR OF DNN USING DIFFERENT INPUT DATA

Input Data	Given Recall Rate					
	95%	90%	85%	80%	75%	70%
G1	23.5	12.2	7.50	5.07	3.45	2.61
RS-G1	21.6	10.9	6.69	4.32	2.85	2.01
RS-G6	20.2	9.57	5.49	3.57	2.31	1.65

IV. EXPERIMENT

Our database contains 63 images (1368 × 972) of the City of San Francisco that were collected from *Google Earth*. A total of 31 images, 3901 vehicles, and 10 3637 samples were used as a training set. The other 32 images, 2870 vehicles, and 115 492 samples were used as a test set.

The false alarm rate (FAR), precision rate (PR), and recall rate (RR) are defined as

$$\begin{cases} \text{FAR} = \frac{\text{number of false alarms}}{\text{number of vehicles}} \times 100\% \\ \text{PR} = \frac{\text{number of detected vehicles}}{\text{number of detected objects}} \times 100\% \\ \text{RR} = \frac{\text{number of detected vehicles}}{\text{number of vehicles}} \times 100\%. \end{cases} \tag{2}$$

A vehicle is detected only if one of its exact locating windows has been detected. All false alarms are fused into small clusters by a small distance limit (20 pixels); the false alarm number is defined as the cluster number.

Table I lists the results of four different methods on our vehicle test set, where the input is only a gray image. Training time is by day. The HOG feature is computed as [15], and its orientation bins is 9. The 48 × 48 gray image is divided into $1 \times 1 + 2 \times 2 + 3 \times 3 + 4 \times 4 + 5 \times 5 = 55$ blocks. The HOG dimension is $55 \times 9 = 495$. The LBP feature is computed as [16], where P = 8, R = 1.5, using 58 uniform patterns and 1 nonuniform pattern. The LBP dimension is $59 \times 55 = 3245$. We utilized the radial basis function kernel and 4000 support vectors in SVM. The kernel parameter is optimized in a range [1/dimension, 40/dimension]. The Adaboost method used the five-type Haar-like features as [18], and 2000 stumps are learned to form the final classifier. DNN training is run in GPU cards, whereas other three methods are run in CPU.

In Table II, G1 is the gray image of the sample. G6 is the six preprocessed images of G1 (refer to Section III-C), RS-G6 is

TABLE III
FAR OF HDNN USING DIFFERENT STRUCTURES

$n_1 - n_2 - n_3$	Given Recall Rate					
	95%	90%	85%	80%	75%	70%
84-00-00	20.2	9.57	5.49	3.57	2.31	1.65
54-20-10	15.0	6.51	3.69	2.34	1.56	1.11
44-20-20	12.8	5.73	3.12	2.01	1.35	0.93
34-30-20	13.8	6.57	3.78	2.49	1.74	1.29
28-28-28	15.5	7.02	4.11	2.61	1.80	1.23
20-44-20	16.3	7.98	4.74	3.00	2.01	1.47
34-10-40	18.3	8.67	4.89	3.12	2.07	1.44

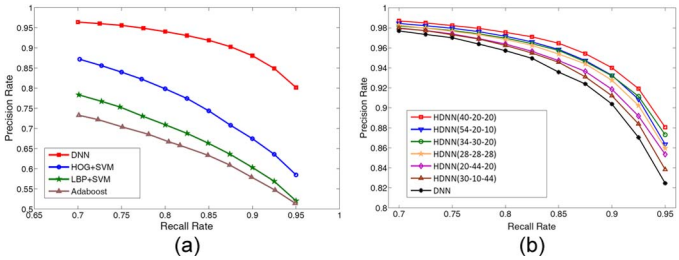

Fig. 6. (a) Recall-Precision-Curves (RPC) of four methods in our vehicle test set; input data is the gray image. (b) RPC of HDNNs and DNN; input data is the combination of six preprocessed images. Here, DNN is just HDNN (84-00-00).

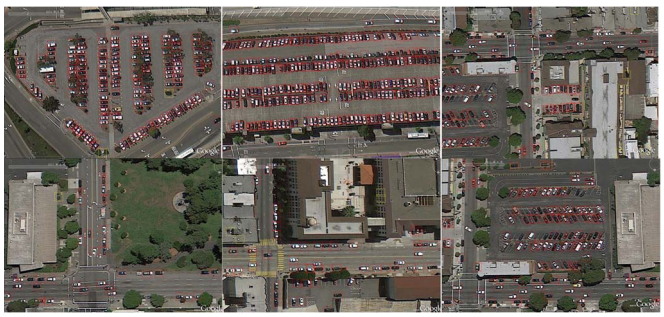

Fig. 7. Some detection results by HDNN (44-20-20) in the City of San Francisco. (Red frames) Cars. (Yellow frames) False alarms.

the rotating and scaling variants of G6. We rotate the G1 sample 11 times by: $0^0, 4.5^0, \ldots, 45^0$, apply 5 scalings to G1 by: 0.8, 0.9, 1.1, 1.2, and 1.3.

In Table III, $n_1 - n_2 - n_3$ denotes the map compose-ratio of the three blocks: B_1, B_2, B_3 (refer to Section III-C). The input data type is RS-G6 (training) and G6 (test). These compose-ratios are deliberately chosen to show the differences of the importance measure between the three blocks. Using the result of HDNN (28-28-28) as the baseline, since the result of HDNN (20-44-20) is worse than that of HDNN (28-28-28), and the result of HDNN (44-20-20) is better, we can deduce that B1 should be more important than B2. The fact that HDNN (34-10-40) is worse than HDNN (20-44-20) proves that B3 has the least importance. HDNN (84-00-00) is just the DNN. Fig. 6(a) shows that DNN outperforms other three methods. Fig. 6(b) shows that HDNN detector performs DNN obviously.

Fig. 7 shows that the HDNN detector performs well in detecting all kinds of vehicles in complex city environments, where the repetitive frames have been fused into the one with the highest classifier's output value.

V. CONCLUSION

Extracting multiscale features is critical to improving the performance of the object detector. However, the state-of-the-art DNN method does not support multiscale feature extraction (the maps of its last max-pooling layer can be considered as the extracted features). We propose HDNNs by dividing all maps of the highest convolutional and max-pooling layer of DNN into multiple blocks of variable receptive field sizes or max-pooling field sizes. We prove that HDNN is capable of extracting multiscale features. Experiments on our vehicle database of the City of San Francisco show that HDNN outperforms DNN obviously.

REFERENCES

[1] T. Zhao and R. Nevatia, "Car detection in low resolution aerial images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, vol. 1, pp. 710–717.
[2] L. Eikvil, L. Aurdal, and H. Koren, "Classification-based vehicle detection in high-resolution satellite images," *J. Photogramm. Remote Sens.*, vol. 64, no. 1, pp. 65–72, Jan. 2009.
[3] P. Liang, G. Teodoro, H. Ling, E. Blasch, G. Chen, and L. Bai, "Multiple Kernel Learning for vehicle detection in wide area motion imagery," in *Proc. 15th Int. Conf. Inf. Fusion*, 2012, pp. 1629–1636.
[4] H. Grabner, T. Nguyen, B. Gruber, and H. Bischof, "On-line boosting-based car detection from aerial images," *J. Photogramm. Remote Sens.*, vol. 63, no. 3, pp. 382–396, May 2008.
[5] A. Kembhavi, D. Harwood, and L. S. Davis, "Vehicle detection using partial least squares," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1250–1265, Jun. 2011.
[6] K. Ali, F. Fleuret, D. Hasler, and P. Fua, "A real-time deformable detector," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 225–239, Feb. 2012.
[7] J. Leitloff, S. Hinz, and U. Stilla, "Vehicle detection in very high resolution satellite images of city areas," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2795–2806, Jul. 2010.
[8] H. Sun, X. Sun, H. Wang, Y. Li, and X. Li, "Automatic target detection in high-resolution remote sensing images using spatial sparse coding bag-of-words model," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 1, pp. 109–113, Jan. 2012.
[9] X. Sun, H. Wang, and K. Fu, "Automatic detection of geospatial objects using taxonomic semantics," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 1, pp. 23–26, Jan. 2010.
[10] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3642–3649.
[11] D. H. Wiesel and T. N. Hubel, "Receptive fields of single neurones in the cats striate cortex," *J. Physiol.*, vol. 148, no. 3, pp. 574–591, Oct. 1959.
[12] K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.
[13] 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.
[14] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, Nov. 2004.
[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 886–893.
[16] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[17] X. Chen, S. Xiang, C. Liu, and C. Pan, "Aircraft detection by deep belief nets," in *Proc. 2nd IAPR Asian Conf. Pattern Recognit.*, Nov. 2013, pp. 54–58.
[18] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.