Detecting pedestrians in surveillance videos based on Convolutional Neural Network and Motion

Domonkos Varga*†, Tamás Szirányi*‡
*MTA SZTAKI, Institute for Computer Science and Control
{varga.domonkos, sziranyi.tamas}@sztaki.mta.hu
†Budapest University of Technology, Department of Networked Systems and Services
‡Budapest University of Technology, Department of Material Handling and Logistics Systems

***Abstract*—Pedestrian detection is a fundamental computer vision task with many practical applications in robotics, video surveillance, autonomous driving, and automotive safety. However, it is still a challenging problem due to the tremendous variations in illumination, clothing, color, scale, and pose. The aim of this paper to present our dynamic pedestrian detector. In this paper, we propose a pedestrian detection approach that uses convolutional neural network (CNN) to differentiate pedestrian and non-pedestrian motion patterns. Although the CNN has good generalization performance, the CNN classifier is time-consuming. Therefore, we propose a novel architecture to reduce the time of feature extraction and training. Occlusion handling is one of the most important problem in pedestrian detection. For occlusion handling, we propose a method, which consists of extensive part detectors. The main advantage of our algorithm is that it can be trained on weakly labeled data, i.e. it does not require part annotations in the pedestrian bounding boxes.**

I. Introduction

Pedestrian detection has been one of the most extensively studied problems in computer vision. One reason is that pedestrian detection is the first step for a number of applications such as smart video surveillance, people-finding for military applications, human-robot interaction, intelligent digital management, and driving assistance system. Pedestrian detection is a rapidly evolving area, as it provides the fundamental information for semantic understanding of the video footages. Because of the various style of clothing in appearance, different possible body articulations, different illumination conditions, the presence of occluding accessories, frequent occlusion between pedestrians, etc., the pedestrian detection is still a challenging problem in computer vision.

The goal of this paper is to present our novel pedestrian detector based on Convolutional Neural Network (CNN) in surveillance videos. In video surveillance, the cameras are usually static and look down to the ground. According to the installing locations, the surveillance scenes can be divided into indoor and outdoor scenes. A surveillance system may contain different modalities and it is beneficial, if a pedestrian detector is able to work both on RGB images and infrared images. In this paper, we propose a novel, dynamic pedestrian detector which has several appealing properties. It contains a set of extensive part detectors that can be trained on weakly labeled data, i.e. it does not require part annotations in the pedestrian bounding boxes. We present an effective CNN architecture to reduce the time of feature extraction and training.

The rest of this paper is organized as follows. In Section II, the related and previous works are reviewed. We describe the proposed pedestrian detector in Section III. Section IV shows experimental results and analysis. We draw the conclusions in Section V.

II. Related Work

There is extensive literature on pedestrian detection algorithms. A deepgoing review on these algorithms is beyond the scope of this paper. We refer readers to comprehensive surveys [1], [2] for more details about existing detectors. We refer to [3] and [4] for older pedestrian detectors. In this section, we review only the works related to our method.

Two representative works in pedestrian detection are the VJ [5] detector and HOG [6] detector. The VJ detector achieved a very fast detection speed with the help of simple Haar-like features and cascade of boosted classifiers. In HOG detector, first, each detection window is decomposed into cells of size 8×8 pixels and each group of 2×2 cells are integrated into a block with an overlap of 50%. A 9-bin histogram of oriented gradients is computed for each cell. Each block is represented by the concatenated histograms of all its cells. This concatenated histogram is normalized to an L2 unit length. Each 128×64 detection window is represented by 15×7 blocks, giving a 3780 dimensional feature vector per detection window. These feature vectors are then used to train a linear SVM classifier. Based on VJ detector, Viola et al. [7] proposed a pedestrian detection system that integrates image intensity information with motion information. A detection style algorithm was used that scans a detector over two consecutive frames of a video sequence. AdaBoost was trained to take advantage of both motion and appearance information to detect a walking pedestrian. Based on HOG, Felzenszwalb et al. proposed the Deformable Part Based Model (DPM) [8] which made a breakthrough in pedestrian detection. The system relies heavily on deformable parts. The authors combined a margin-sensitive approach for data mining hard negative examples with latent SVM. Latent SVM is semi-convex [9], but the training problem becomes convex once latent information is specified for the positive examples.

Since the feature extraction pipelines in above mentioned methods are designed manually, the literature categorizes them

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

as hand-craft features. In recent years, deep learning techniques have achieved amazing success in modeling large-scale data. ConvNet [10] uses a mix of unsupervised and supervised training to create a deep convolutional neural network trained on INRIA pedestrian dataset. Chen et al. [11] proposed to cascade simple Aggregated Channel Features (ACF) and rich Deep Convolutional Neural Network features in order to detect pedestrians in complex scenes. The ACF based detector was used to rapidly generate candidate pedestrian windows and the rich DCNN features with a SVM was used for fine classification.

Another line of work focuses on using deep architectures to jointly model parts and occlusions. Ouyang et al. [12] conducted Restricted Boltzmann Machine (RBM) in modeling mutual visibility relationship for occlusion handling. In [13], the authors proposed a joint deep learning framework and a new deep network architecture that jointly learns feature extraction, deformation handling, occlusion handling, and classification.

Motivated by these previous works, in the rest of the paper, we propose a novel, dynamic pedestrian detector based on CNN with occlusion handling.

III. Our system architecture

Figure 1 presents the overview of our dynamic pedestrian detector. In order to integrate image intensity information with motion information, the absolute difference of two consequtive video frames is computed. The pattern of human motion is well known to be readily distinguishable from other sorts of motion. A detection window scans the absolute difference of the two frames and a pre-trained CNN classifier determines for each detection window independently whether it contains a pedestrian or not. In many applications such as video surveillance, detection speed is as important as accuracy. Although the CNN has good generalization power, training and applying of CNN classifier is time-consuming.

The main advantage of CNN that it integrates feature extraction and classification into one single structure, however both process with training can be very time consuming. In order to reduce the time of feature extraction and classification process of the CNN, we present a not fully connected arhitecture. However, the rarefication of the connections can degrade the generalization power of the CNN, we load motion patterns to the input of CNN, which is well known to be readily distinguishable from other sorts of motion and in that way support the training process. To figure out an optimal connection, it is important to reduce the training time. In Subsection III-B, we describe our training method that results in significant fall-of of the training time compared to [14]. Finally, we propose a occlusion handling method which appealing property is that it can be trained on weakly labeled data.

A. Architecture of CNN

A CNN contains usually three types of layers: convolutional layers, sub-sampling layers, and a neuron layer. These types

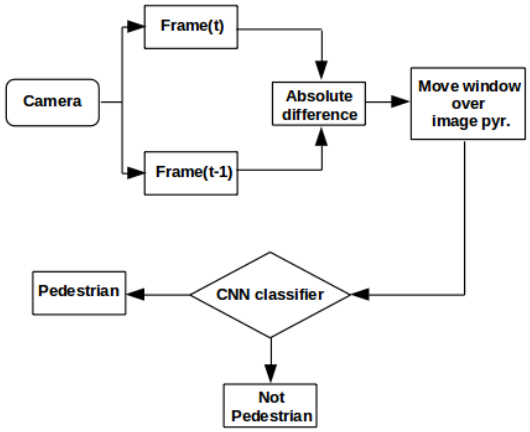

Fig. 1. Architecture of our pedestrian detection system.

of layers are arranged in a feed-forward structure. A convolutional layer is followed by a sub-sampling layer, and the last convolutional layer is followed by the neuron layer.

The output of convolutional layer l is calculated as [15]

$$\mathbf{y}_n^l = \sigma_l\Big(\sum_{m\in\mathbf{V}_n^l} \mathbf{y}_m^{l-1} \otimes \mathbf{w}_{m,n}^l + b_n^l\Big), \tag{1}$$

where $\otimes$ denotes the 2-D convolution operator, $\mathbf{y}_m^{l-1}$ are the input feature maps, $\mathbf{w}_{m,n}^l$ stands for the convolution mask from feature map m in layer $l-1$ to feature map n in layer l, b_n^l is the bias term associated with feature map n, $\mathbf{V}_n^l$ stands for the list of all planes in layer $l-1$ that are connected to feature map n, $\sigma()$ is the activation function, and $\mathbf{y}_n^l$ is the output feature map. If the size of the input feature maps is $H \times W$ and the size of the convolution mask is $h \times w$, then the size of the output feature map will be $(H-h+1)\times(W-w+1)$.

A sub-sampling layer follows through a form of non-linear down-sampling. The sub-sampling layer partitions the input feature map into a set of non-overlapping rectangles and, for each sub-region, outputs the maximum, the minimum, or the average. If the sub-sampling layer divides an input feature map size of $H\times W$ into non-overlapping blocks of size 2×2 pixels, then the size of the output feature map will be $(H/2)\times(W/2)$.

Lets consider that the neuron layer l contains N neurons. The output of neuron n in the layer is computed as [15]

$$y_n^l = \sigma^l\Big(\sum_{m=1}^{N} y_m^{l-1} w_{m,n}^l + b_n^l\Big), \tag{2}$$

where b_n^l denotes the bias term of neuron n in layer l, $w_{m,n}^l$ stands for the weight from feature map m to feature map n in layer l, and $\sigma()$ is the activation function.

Table I defines exactly the applied CNN architecture. It consists of three convolutional layer, two sub-sampling layers, and a neuron layer. We define the connections between the layer with the help of a list that contains all feature maps in layer $l-1$ that are connected to feature map n in layer l. For instance, $\mathbf{V}_2^5 = \{2,3,4\}$ represents that feature maps $2,3,4$ in layer 4 are connected to feature map 2 in layer 5.

The size of the filter in `conv1` is 9×7, in `sub-sampling1` is 3×3, in `conv2` is 7×7, in `sub-sampling2` is 3×3, and in `conv3` is 10×7, respectively. In the sub-sampling layers, we applied min pooling. The activation functions are linear for convolutional layers and sigmoidal for sub-sampling layers and neuron layer.

TABLE I
THE STRUCTURE OF THE APPLIED CONVOLUTIONAL NEURAL NETWORK. $\mathbf{V}_n^l$ STANDS FOR THE LIST OF ALL PLANES IN LAYER $l-1$ THAT ARE CONNECTED TO FEATURE MAP n IN LAYER l. THE SIZE OF THE INPUT PATTERNS ARE 80×60.

Type	Output map size	Connection
`conv1`	4@72 × 54	$\mathbf{V}_1^1 = \{1\}$, $\mathbf{V}_2^1 = \{1\}$, $\mathbf{V}_3^1 = \{1\}$, $\mathbf{V}_4^1 = \{1\}$
`sub-sampling1`	4@24 × 18	$\mathbf{V}_1^2 = \{1\}$, $\mathbf{V}_2^2 = \{2\}$, $\mathbf{V}_3^2 = \{3\}$, $\mathbf{V}_4^2 = \{4\}$
`conv2`	6@18 × 12	$\mathbf{V}_1^3 = \{1\}$, $\mathbf{V}_2^3 = \{2\}$, $\mathbf{V}_3^3 = \{1,3\}$, $\mathbf{V}_4^3 = \{2,4\}$, $\mathbf{V}_5^3 = \{3\}$, $\mathbf{V}_6^3 = \{4\}$
`sub-sampling2`	6@6 × 4	$\mathbf{V}_1^4 = \{1\}$, $\mathbf{V}_2^4 = \{2\}$, $\mathbf{V}_3^4 = \{3\}$, $\mathbf{V}_4^4 = \{4\}$, $\mathbf{V}_5^4 = \{5\}$, $\mathbf{V}_6^4 = \{6\}$
`conv3`	5@1 × 1	$\mathbf{V}_1^5 = \{1,2\}$, $\mathbf{V}_2^5 = \{2,3,4\}$, $\mathbf{V}_3^5 = \{3,4\}$, $\mathbf{V}_4^5 = \{4,5\}$, $\mathbf{V}_5^5 = \{5,6\}$
`neuron layer`	2@1 × 1	$\mathbf{V}_1^6 = \{1,2,3,4,5\}$, $\mathbf{V}_2^6 = \{1,2,3,4,5\}$

B. Training of CNN

We created a set of video sequences of street and indoor scenes with all pedestrians marked with a bounding box. We have six such sequences, each containing around 2500 frames. We used these sequences to create a training set from which we learned our dynamic pedestrian detector. The detector was trained on consecutive pedestrian image pairs. We have collected 6,000 consecutive pedestrian image pairs and we also collected 10,000 negative examples. All positive and negative examples have been aligned and scaled to the dimensions 80×60.

The model parameters of the CNN are initialized randomly with Gaussian distribution. Given a dataset of M input examples $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M)$, the CNN is trained to give target outputs $(\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_M)$, where $\mathbf{d}_i$ is set to $(+0.9, 0)$ if $\mathbf{x}_i$ a pedestrian motion pattern; otherwise $\mathbf{d}_i$ is set to $(0, -0.9)$. We divided our $\mathbf{X}$ training set into smaller subsets $\mathbf{X} = \mathbf{X}_1 \bigcup \mathbf{X}_2 \bigcup ... \bigcup \mathbf{X}_N$, and compute the error gradient for each separate subset: $g_1(\mathbf{w}), g_2(\mathbf{w}), ..., g_N(\mathbf{w})$, where $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_L)$ stands for all weights and biases of the network. Training is done in each subset iteratively to minimize the mean-square-error function:

$$L_j(\mathbf{w}) = \frac{1}{M_j} \sum_{i=1}^{M_j} (\mathbf{y}_i - \mathbf{d}_i)^2, j = 1, 2, ..., N, \quad (3)$$

where $\mathbf{y}_i$ is the output of the CNN. The error gradient of each subset can be computed as $g_j(\mathbf{w}) = \bigtriangledown L_j(\mathbf{w})$. The overall error gradient vector can be computed as

$$g(\mathbf{w}) = \sum_{j=1}^{N} g_j(\mathbf{w}). \quad (4)$$

After the error gradient vector is computed, Levenberg-Marquardt [14] back-propagation algorithm is applied in order to train the network. The Levenberg-Marquardt algorithm is a trust-region method that uses the Gauss-Newton approximation of the Hessian matrix. From second-order Taylor expansion and Gauss-Newton approximation of Hessian matrix:

$$\Delta \mathbf{w}(t) = -[\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \bigtriangledown E, \quad (5)$$

$\mathbf{J}$ is the Jacobian matrix, $\bigtriangledown E$ is determined through the Jacobian matrix $\mathbf{J}$, and μ represents an adaptive parameter controlling the size of the trust region. The details of the algorithm can be found in the given reference.

We have trained 45 network structure with different connections and activation functions using the presented technique of this subsection. After training, all the networks were evaluated on a validation video, and we selected the best network. The parameters and the connections of the best one was described in the previous subsection. The training time for each network structure was about 12 hours of training on one personal computer. Using the classical Levenberg-Marquardt, the training process took about 48 hours on the same computer.

C. Occlusion Handling

In our occlusion handling method, we determine first whether the score of the holistic classifier is ambiguous. When the output is ambiguous, an occlusion inference process is applied (Fig. 2).

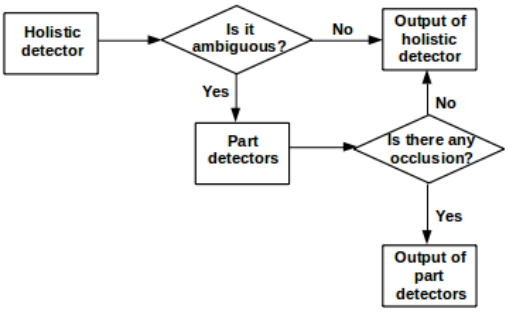

Fig. 2. Occlusion handling scheme.

We consider pedestrian's motion pattern as a rigid object and define a grid of $M_1 \times M_2$, where M_1 and M_2 indicate the numbers of cells in horizontal and vertical direction, respectively. Each cell is a square and has equal size. We ensure each part to be a rectangle. The possible sizes of the parts can be defined as

$$S = \{(w, h) \mid W_{min} \leq w \leq M_1, H_{min} \leq h \leq M_2, w, h \in \mathbb{N}^+\}, \quad (6)$$

where w and h stand for the width and height of a part in terms of the number of cells they contain. W_{min} and H_{min}

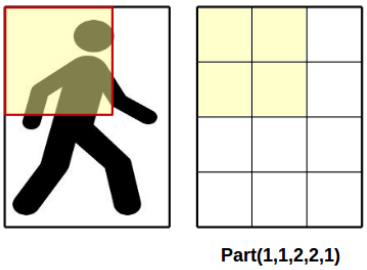

Fig. 3. Part prototype example, (x, y, w, h, i) is defined in Eq. 7. The head-shoulder part with 2 grids in height and 2 grids in width.

are used to avoid subtle parts. Then, for each $(w, h) \in S$, we slide a $w \times h$ window over the grid to generate parts at different positions. The entire part pool can be defined as follows

$$P = \{(x, y, w, h, i) \mid x, y \in \mathbb{N}^+, (w, h) \in S, i \in I\}, \quad (7)$$

where x and y stand for the coordinates of the top-left cell in the part and i is a unique id. For instance, the part representing the full motion pattern is defined as $(1, 1, M_1, M_2, I_1)$, for another example see Figure 3. Because of our training examples are in size of 80×60, we have used in our implementation the following parameters $M_1 = 3$, $M_2 = 4$, $W_{min} = 2$, $H_{min} = 2$, and the step size is one.

For each part, a similar CNN was trained. If the output of the holistic detector is ambiguous, we run the part detectors. We take only into account the results of the part detectors with the four highest scores.

IV. Experimental results

We perform the experiments on CAVIAR sequences (`http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1`), which is captured in a corridor with resolution 384×288 pixels. In this paper, we use per-image performance, plotting detection rate versus false positives per-image (FPPI). Figure 4 shows some sample detections on the CAVIAR sequences. Figure 5 shows the detection rate versus false positive per-image (FPPI) for the presented detector and six other systems. The six other systems we compare include Dalal and Triggs HOG+SVM system [6], Lie et al. HOG+AdaBoost system [16], Papageorgiou et al. Haar+SVM system [17], Monteiro et al. Haar+AdaBoost system [18], a PHOG+HIKSVM system [19], and a system based on Aggregated Channel Features [20]. Table II summarizes the speed comparison, we remark that our source code is still not optimized while the source code of the others are mainly optimized.

In order to prove the generalization performance of our system, we applied our algorithm to the video frames of an infrared surveillance camera. Our presented system mainly captures the pedestrian's motion pattern, that is why shows high invariance to illumination and clothing, and performs well in infrared images too. Figure 6 shows some sample detections in infrared images.

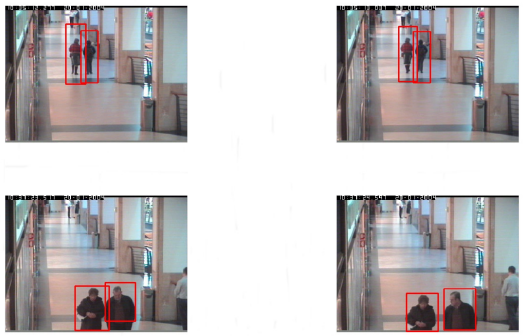

Fig. 4. Some detections on CAVIAR sequences.

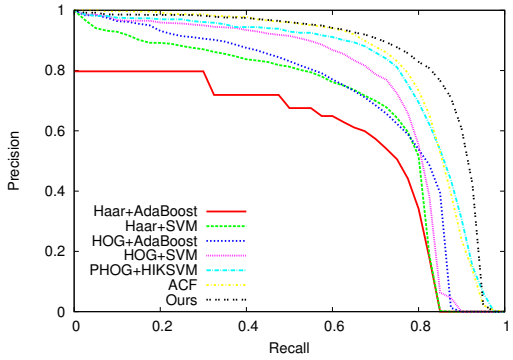

Fig. 5. Detection rate versus false positive per-image (FPPI) curves for pedestrian detectors. 2×2 is the step size and 1.09 is the scale factor of the sliding-window detection.

Fig. 6. Some detections on infrared images.

V. Conclusion

In this paper, we presented a novel, dynamic pedestrian detection system and reported on experimental results. We have presented our dynamic pedestrian detector based on CNN. We have also shown that the system is able to detect pedestrians in real-time in surveillance videos. All the modules of the algorithm were introduced with its details about design and implementation. We have compared the system to other algorithms and presented results on RGB images and thermal images. We have proved that our detector is highly invariant to clothing and illumination. Our experiments also demonstrated

TABLE II
SPEED COMPARISON.

Method	Speed
Haar+AdaBoost [18]	15.63 fps
Haar+SVM [17]	13.56 fps
HOG+AdaBoost [16]	9.48 fps
HOG+SVM [6]	4.27 fps
PHOG+HIKSVM [19]	6.19 fps
ACF [20]	14.03 fps
Ours	5.48 fps

that the pedestrian detector is able to provide a robust input for a surveillance system and it can work on different modalities.

There are many directions for further research. It is worth studying how to extract a more distinguishable pedestrian motion pattern in order to improve the performance of the classifier. Another further direction of research is improving the searching strategy of the detection window. To make our system faster, we would like to apply parallel architectures. However, our detector cannot handle the articulated deformation of pedestrians, which is the next problem to be attacked.

ACKNOWLEDGMENT

This work has been supported by the EU FP7 Programme (FP7-SEC-2011-1) No. 285320 (PROACTIVE project). The research was also partially supported by the Hungarian Scientific Research Fund (No. OTKA 106374). The research was also supported by Bosch ERNYO 13-1-2013-0007 project.

REFERENCES

[1] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *Computer Vision-ECCV 2014 Workshops*. Springer, 2014, pp. 613–627.

[2] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 304–311.

[3] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 11, pp. 1863–1868, 2006.

[4] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 12, pp. 2179–2195, 2009.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[7] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[9] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[10] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3626–3633.

[11] X. Chen, P. Wei, W. Ke, Q. Ye, and J. Jiao, "Pedestrian detection with deep convolutional neural network," in *Computer Vision-ACCV 2014 Workshops*. Springer, 2014, pp. 354–365.

[12] W. Ouyang, X. Zeng, and X. Wang, "Modeling mutual visibility relationship in pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3222–3229.

[13] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2056–2063.

[14] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *Neural Networks, IEEE Transactions on*, vol. 5, no. 6, pp. 989–993, 1994.

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[16] G. Lie, G. Ping-shu, Z. Yi-bing, Z. Ming-heng, and L. Lin-hui, "Pedestrian detection based on hog features optimized by gentle adaboost in roi." *Journal of Convergence Information Technology*, vol. 8, no. 2, 2013.

[17] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[18] G. Monteiro, P. Peixoto, and U. Nunes, "Vision-based pedestrian detection using haar-like features," *Robotica*, vol. 24, pp. 46–50, 2006.

[19] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[20] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 8, pp. 1532–1545, 2014.