Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection

Wanli Ouyang, *Senior Member, IEEE,* Hui Zhou, *Student Member, IEEE*, Hongsheng Li, *Member, IEEE* ,Quanquan Li *Member, IEEE*, Junjie Yan, *Member, IEEE*, Xiaogang Wang *Member, IEEE*

Abstract—Feature extraction, deformation handling, occlusion handling, and classification are four important components in pedestrian detection. Existing methods learn or design these components either individually or sequentially. The interaction among these components is not yet well explored. This paper proposes that they should be jointly learned in order to maximize their strengths through cooperation. We formulate these four components into a joint deep learning framework and propose a new deep network architecture (Code available on www.ee.cuhk.edu.hk/~wlouyang/projects/ouyangWiccv13Joint/index.html). By establishing automatic, mutual interaction among components, the deep model has average miss rate 8.57%/11.71% on the Caltech benchmark dataset with new/original annotations.

Index Terms—CNN, convolutional neural networks, object detection, deep learning, deep model

1 Introduction

Pedestrian detection is a key technology in automotive safety, robotics, and intelligent video surveillance. It has attracted a great deal of research interest [3], [9], [17], [70], [12]. The main challenges of this task are caused by the intra-class variation of pedestrians in clothing, lighting, backgrounds, articulation, and occlusion.

In order to handle these challenges, a group of inter-dependent components are important. First, features should capture the most discriminative information of pedestrians. Well-known features such as Haar-like features [72], SIFT [37], and HOG [9] are designed to be robust to intra-class variation while remain sensitive to inter-class variation. Recently, deeply learned features are found to be effective in pedestrian detection and generic object detection [82], [81], [49], [54], [66], [26], [88]. Second, deformation models should handle the articulation of human parts such as torso, head, and legs. The state-of-the-art deformable part-based model in [22] allows human parts to articulate with constraint. Third, occlusion handling approaches [18], [76], [24], [42] seek to identify the occluded regions and avoid their use when determining the existence of a pedestrian in a window. Finally, a classifier decides whether a candidate window shall be detected as enclosing a pedestrian. SVM [9], boosted classifiers [16], random forests [14], and their variations are often used.

Although these components are interdependent, their interactions have not been well explored. Currently, they are first learned or designed individually or sequentially, and then put together in a pipeline. The interaction among these components is usually achieved using manual parameter configuration. Consider the following three examples. (1) The HOG feature is individually designed with its parameters manually tuned given the linear SVM classifier being used in [9]. Then the HOG feature is fixed when people design new classifiers [40]. (2) A few HOG feature parameters are tuned in [22] and fixed, and then different part models are learned in [22], [93]. (3) By fixing HOG features and deformable models, occlusion handling models are learned in [46], [50], using the part-detection scores as input.

As shown in Fig. 1, the motivation of this paper is to establish automatic interaction in learning these key components. We hope that jointly learned components, like members with team spirit, can create synergy through close interaction, and generate performance that is greater than individually learned components. For example, well-learned features help to locate parts, meanwhile, well-located parts help to learn more discriminative features for different parts. This paper formulates the learning of these key components into a unified deep learning problem. The deep model is especially appropriate for this task because it can organize these components into different layers and jointly optimize them through back-propagation.

This paper makes the following three main contributions.

1) A unified deep model for jointly learning feature extraction, a part deformation model, an occlusion model and classification. With the deep model, these components interact with each other in the learning process, which allows each component to maximize its strength when cooperating with others.

• *Wanli Ouyang, Hui Zhou, Hongsheng Li, Quanquan Li, Xiaogang Wang are with the Department of Electronic Engineering at the Chinese University of Hong Kong, Hong Kong. Junjie Yan is with the SenseTime Group Limited. Wanli Ouyang and Xiaogang Wang are corresponding authors. E-mail: wlouyang, xgwang@ee.cuhk.edu.hk.*

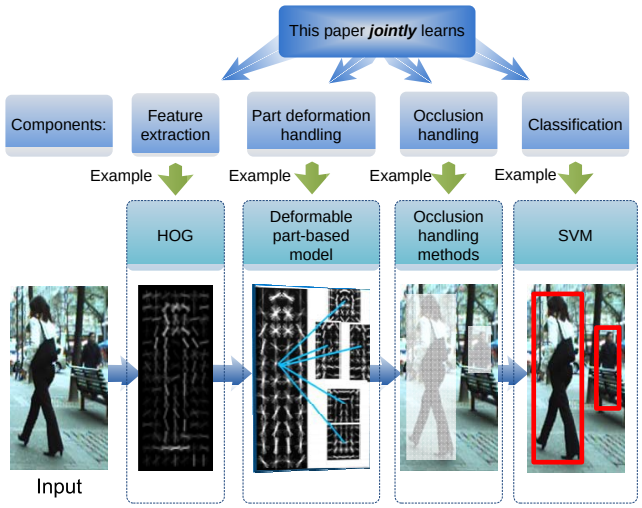


Figure 1. Motivation of this paper to jointly learn the four key components in pedestrian detection: feature extraction, deformation handling models, occlusion handling models, and classifiers.

2) We enrich the operation in deep models by incorporating the deformation layer into the convolutional neural networks (CNN) [35]. With this layer, various deformation handling approaches can be applied to our deep model.
3) The features are learned from pixels through interaction with deformation and occlusion handling models. Such interaction helps to learn more discriminative features.

2 Related Work

It has been proved that deep models are potentially more capable than shallow models in handling complex tasks [5]. They have achieved spectacular progress in computer vision [28], [29], [60], [32], [34], [45], [33], [86], [39], [67], [21], [58], [81], [82], [74], [75], [92]. Deep models for pedestrian detection focus on feature learning [64], [45], contextual information learning [87], and occlusion handling [46]. Recent reviews and performance evaluations are provided in [4], [17].

Many features are utilized for pedestrian detection. Haar-like features [72], [89], HOG [9], and dense SIFT [71] are designed to capture the overall shape of pedestrians. First-order color features like color histograms [16], various channel features [91], [16], [13], second-order color features like color-self-similarity (CSS) [73] and co-occurrence features [63] are also used for pedestrian detection. Texture feature like LBP are used in [76]. Other types of features include the covariance descriptor [70], depth [20], segmentation results [18], 3D geometry [30], motion [57], and their combinations [36], [76], [16], [73], [18], [63]. All the features mentioned above are designed manually. Recently, researchers have become aware of the benefit of learning features from training data [2], [45], [64], [82], [31], [38], [69], [68]. Similar to HOG, they use local max pooling or average pooling to be robust to small local misalignment. However, these approaches do not learn the variable deformation properties of body parts. The approaches in [11], [8] learn features and a part-based model sequentially but not jointly.

Since pedestrians have non-rigid deformation, the ability to handle deformation improves detection performance. Deformable part-based models are used in [22], [93], [56], [48], [51], [80] for handling translational movement of parts. To handle more complex articulations, size change and rotation of parts are modeled in [23], and mixture of part appearance and articulation types are modeled in [6], [83], [10]. In these approaches, features are manually designed.

Recently, the learning of deformation from convolutional layers is presented in [47], [27], [49]. First, par visibility learning is learned in our approach but not learned in these approaches. Second, these approaches can only place the deformation handling after the convolutional layers, which restricts the deep architecture especially for AlexNet [33], ZF-Net [85] and VGG [66] with fully connected layers. We provide a CNN design and show that deformation handling can be used even for the models with fully connected layers by treating them as the filter for full body.

In order to handle occlusion, many approaches have been proposed for estimating the visibility of parts [18], [76], [79], [78], [65], [36], [68], [52], [53]. Some of them use the detection scores of blocks or parts [76], [46], [18], [79], [68] as input for visibility estimation. The approach in [42] handles occlusion by designing multiple classifiers. Some use other cues like segmentation results [36], [18] and depth [18]. However, all these approaches learn the occlusion modeling separately from feature extraction and part models.

The widely used classification approaches include various boosting classifiers [14], [16], [78], linear SVM [9], histogram intersection kernel SVM [40], latent SVM [22], multiple kernel SVM [71], structural SVM [93], and probabilistic models [3], [43]. In these approaches, classifiers are adapted to training data, but features are designed manually. If useful information has been lost at feature extraction, it cannot be recovered during classification. Ideally, classifiers should guide feature learning.

In summary, previous works treat the components individually or sequentially. This paper takes a global view of these components and is an important step towards joint learning of them for pedestrian detection.

3 The Basic Deep Model

3.1 Overview

An overview of our examplar deep model is shown in Fig. 2. In this model:

1) *Input image data* are obtained by warping the candidate box into 3 channels.
2) *Feature maps* are obtained by a convolutional layer and its following average pooling layer. The 3-channel input image data is convolved with $9 \times 9 \times$ filters and outputs 64 maps. $|tanh(x)|$, i.e. activation function $tanh$ and absolution value rectification, is used for each filter response x. Then the 64 filtered data maps goes through the average pooling layer using 4×4 boxcar filters with a 4×4 subsampling step.
3) *Part detection maps* are obtained from the second convolutional layer. This layer convolves the feature maps with 20

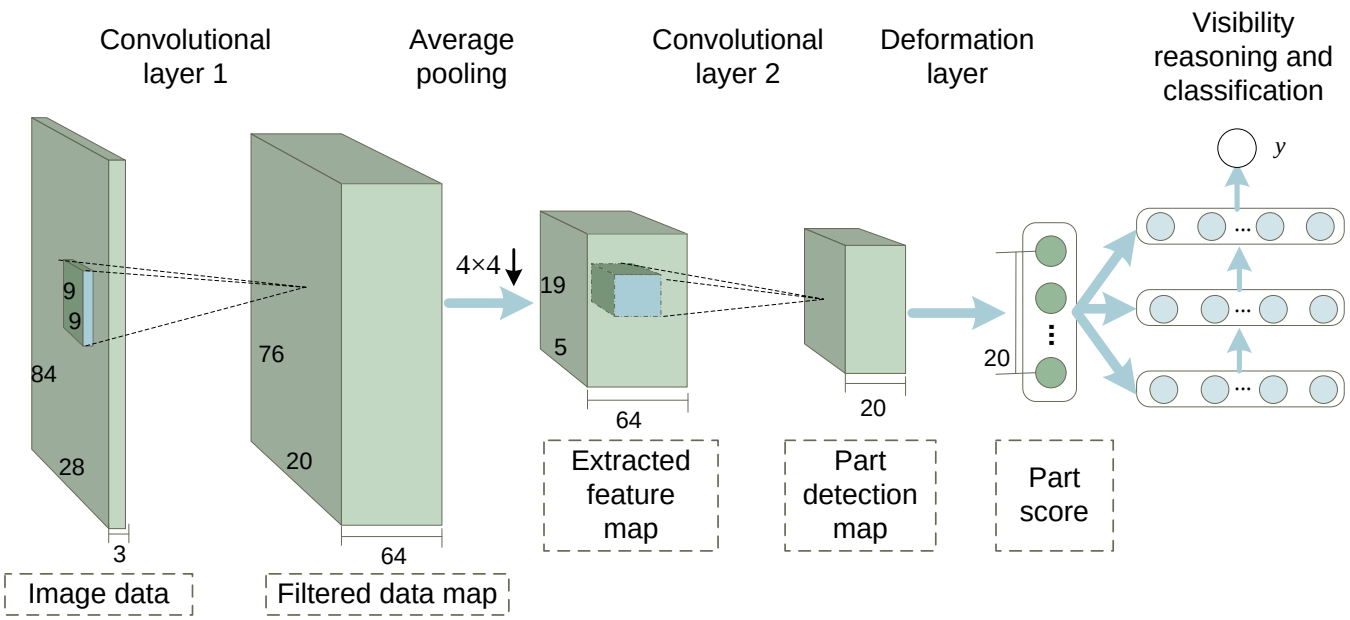


Figure 2. Overview of our basic deep model. Image data is convolved with 64 $9 \times 9 \times 3$ filters and averagely pooled to obtain 64 feature maps. The feature maps are then processed by the second convolutional layer and the deformation layer to obtain 20 part scores. Finally the visibility reasoning model is used to estimate the detection label y.

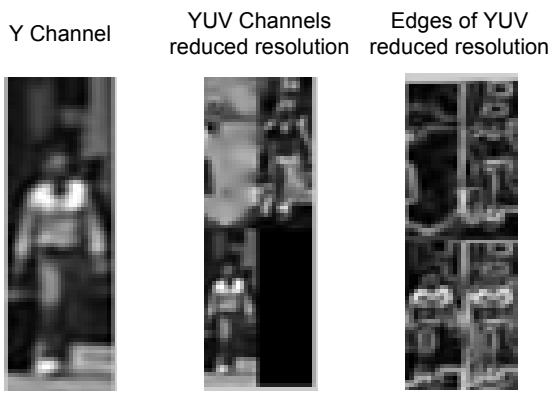


Figure 3. Preparation of three-channel input data, i.e. 84×28 Y-channel image (left), concatenation of 42×14 YUV channels (right), and concatenation of the edges for 42×14 YUV image.

part filters of different sizes and outputs 20 part detection maps. Details are given in Section 3.3.

4) *Part scores* are obtained from the 20 part detection maps using a deformation modeling layer. This layer outputs 20 part scores. Details are given in Section 3.4.
5) The visibility reasoning of 20 parts is used for estimating the label y; that is, whether a given window encloses a pedestrian or not. Details are given in Section 3.5.

At the training stage, all the parameters are optimized through back-propagation (BP).

3.2 Input data preparation

The detection windows of different sizes are warped into images with height 84 and width 28, in which pedestrians have height 60 and width 20. Then the 3-channel input data for CNN are obtained as follows:

(1) The first channel is a 84×28 Y-channel image after the image is converted into the YUV color space.

(2) The 42×14 images in the YUV color space are concatenated into the second channel of size 84×28 with zero padding. The 42×14 images are warped from the 84×28 images.

(3) Four 42×14 edge maps are concatenated into the third channel of size 84×28. Three edge maps are obtained from the three-channel 42×14 image in the YUV color space. The magnitudes of horizontal and vertical edges are computed using the Sobel edge detector. The fourth edge map is obtained by choosing the maximum magnitudes from the first three edge maps.

Fig. 3 shows the three channels for a pedestrian patch. In this way, information about pixel values at different resolutions and information of primitive edges are utilized as the input of the first convlutional layer to extract features. The first convolutional layer and its following average pooling layer use the standard CNN settings.

We empirically find that it is better to arrange the images and edge maps into three concatenated channels instead of eight separate channels. In order to deal with illumination change, the data in each channel is preprocessed to be zero mean and unit variance.

3.3 Generating the part detection map

Normally, the filter size of a convolutional layer is fixed [35], [33]. Since the parts of pedestrians have different sizes, we design the filters in the convolutional layer with variable sizes when obtaining part detection maps. As shown in Fig. 4(a), we design parts at three levels with different sizes. There are six small parts at level 1, seven medium-sized parts at level 2, and seven large parts at level 3, as shown in Fig. 4(a).

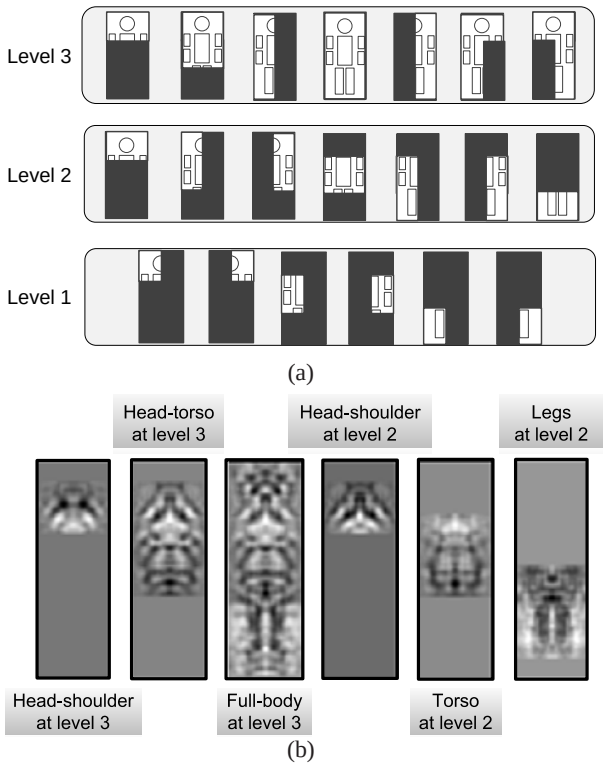


Figure 4. The parts model (a) and the filters (b) learned at the second convolutional layer. We follow [19] and visualize the filter that optimizes the corresponding stimuli of the neurons, which is also used in [34].

A part at an upper level is composed of parts at the lower level. Parts at the top level are also the possible occlusion statuses. Gray color indicates occlusion. The other two levels are body parts. In the figure, the head-shoulder part appears twice (representing occlusion status at the top level and part at the middle level respectively) because this body part itself can generate an occlusion status. Fig. 4(b) shows a few part filters learned with our deep model. They are visualized using the activation maximization approach in [19]. The figure shows that the head-shoulder at level 2 and the head-shoulder at level 3 extract different visual cues from the input image. The head-shoulder filters in Fig. 4(b) contain more detailed silhouette information on heads and shoulders than the head-shoulder filter learned with HOG in Fig. 1. The two-legs filter in Fig. 4(b) is visually more meaningful than the one learned with HOG in Fig. 1.

3.4 The deformation layer

In order to learn the deformation constraints of different parts, we propose the deformation handling layer (deformation layer for short) for the CNNs.

Denote pth part detection map by $\mathbf{M}_p, p = 1, \ldots, P$. The deformation layer takes the P part detection maps as input and outputs P part scores $\mathbf{s} = \{s_1, \ldots, s_P\}$, $P = 20$ in Fig. 2. The deformation layer treats the part detection maps separately and produces the pth part score s_p from the p part detection map $\mathbf{M}_p$. A 2D summed map, denoted by $\mathbf{B}_p$, is obtained by summing up the part detection map $\mathbf{M}_p$ and the deformation maps as follows:

$$\mathbf{B}_p = \mathbf{M}_p - \sum_{n=1}^{N} c_{n,p} \mathbf{D}_{n,p}. \quad (1)$$

$\mathbf{D}_{n,p}$ denotes the nth deformation map for the pth part, $c_{n,p}$ denotes the weight for $\mathbf{D}_{n,p}$, and N denotes the number of deformation maps. s_p is globally max-pooled from $\mathbf{B}_p$ in Eq. (1):

$$s_p = \max_{(x,y)} b_p^{(x,y)}, \quad (2)$$

where $b_p^{(x,y)}$ denotes the (x, y)th element of $\mathbf{B}_p$. The detected part location can be inferred from the summed map as follows:

$$(\tilde{x}_p, \tilde{y}_p) = \arg\max_{(x,y)} b_p^{(x,y)}. \quad (3)$$

The $c_{n,p}$ and $\mathbf{D}_{n,p}$ in (1) are the keys for designing different deformation models. Both $c_{n,p}$ and $\mathbf{D}_{n,p}$ can be considered as the parameters to be learned. Three examples are given below.

3.4.1 Example 1

Suppose $N = 1$, $c_{1,p} = 1$ and the deformation map $\mathbf{D}_{1,p}$ is the parameter to be learned. In this case, the discrete locations of the pth part are treated as bins and the deformation cost for each bin is learned. $d_{1,p}^{(x,y)}$, which denotes the (x, y)th element of $\mathbf{D}_{1,p}$, corresponds to the deformation cost of the pth part at location (x, y). For $\mathbf{D}_{1,p}$ of size $H_D \times W_D$, there are $H_D \cdot W_D$ parameters to be learned. The approach in [59] treats deformation cost in this way.

3.4.2 Example 2

$\mathbf{D}_{1,p}$ can also be predefined. Suppose $N = 1$ and $c_{n,p} = 1$. If $d_{1,p}^{(x,y)}$ is the same for any (x, y), then there is no deformation cost. If $d_{1,p}^{(x,y)} = \infty$ for $(x, y) \notin \mathbb{X}$, $d_{1,p}^{(x,y)} = 0$ for $(x, y) \in \mathbb{X}$, then the parts are only allowed to move freely in the location set $\mathbb{X}$. Max-pooling is a special case of this example by setting $\mathbb{X}$ to be a local region. The disadvantage of max-pooling is that the hand-tuned local region $\mathbb{X}$ does not adapt to different deformation properties of different parts.

3.4.3 Example 3

The deformation layer can represent the widely used quadratic constraint of deformation in the deformable part model (DPM) [22].

DPM. An object hypothesis specifies the location of each part in the model $\mathbf{l} = (l_1, \ldots, l_P)$, where $l_p = (x_p, y_p)$ specifies the location of the pth part. The root part corresponds to $p = 1$. In DPM, the score of the pth part is given by its part detection score at location l_p (the data term) minus a deformation cost that depends on the relative position of pth part with respect to the root (the spatial prior) as follows:

$$score(l_p) = s(I, l_p) - < \mathbf{c}_p, \psi(l_1, l_p) >, \quad (4)$$

where $s(I, l_p)$ (the data term) denotes the pth part detection score at location l_p for image I. The spatial prior in (4) is modeled as follows:

$$\begin{aligned} < \mathbf{d}_p, \psi(l_1, l_p) > = & c_{1,p}(x_p - x_1 - a_{x,p})^2 + c_{2,p}(y_p - y_1 - a_{y,p})^2 \\ & + c_{3,p}(x_p - x_1 - a_{x,p}) + c_{4,p}(y_p - y_1 - a_{y,p}), \end{aligned} \quad (5)$$

where $c_{i,p}$ for $i = 1, 2, 3, 4$ are parameters to be learned, (x_p, y_p) are locations of the pth part with (x_1, y_1) being the root, $(a_{x,p}, a_{y,p})$ denotes the relative anchor location between the pth part and the root.

Relationship between the DPM and the deformation layer. Below, we skip the subscript $_p$ used in (1)-(5) to be concise. The deformation layer in (1) can be represented as follows:

$$\begin{aligned}
&\mathbf{B}_p = \mathbf{M}_p + \sum_{n=1}^{4} c_{n,p}\mathbf{D}_{n,p},\\
&b_p^{(x_p,y_p)} = m_p^{(x_p,y_p)} + \sum_{n=1}^{4} c_{n,p} d_{n,p}^{(x_p,y_p)},\\
&d_{1,p}^{(x_p,y_p)} = (x_p - x_1 - a_{x,p})^2, \quad d_{2,p}^{(x_p,y_p)} = (y_p - y_1 - a_{y,p})^2,\\
&d_{3,p}^{(x_p,y_p)} = x_p - x_1 - a_{x,p}, \quad d_{4,p}^{(x_p,y_p)} = y_p - y_1 - a_{y,p},
\end{aligned} \tag{6}$$

where $d_{n,p}^{(x,y)}$ is the (x,y)th element of $\mathbf{D}_{n,p}$ and $m^{(x,y)}$ is the (x,y)th element of the part detection map $\mathbf{M}_p$. The $m^{(x,y)}$ in (6) corresponds to the data term $s(I, l_p)$ (4) and the $\sum_{n=1}^{4} c_{n,p}\mathbf{D}_{n,p}$ in (6) corresponds to the spatial prior of DPM in (5). In DPM [22], the part score $s(I, l_p)$ for all locations $l_p \in \mathbb{L}$ is obtained by convolving HOG features with part filters. In our CNN, the part detection score $m^{(x,y)}$ is obtained by convolving CNN features with CNN part features as illustrated in Section 3.3. c_1, c_2, c_3 and c_4 are parameters to be learned and $\mathbf{D}_n$ for $n = 1, 2, 3, 4$ are predefined according to the pre-defined anchor (a_x, a_y) and root location (x_1, y_1). Fig. 5 illustrates this example. There is no deformation cost if $c_n = 0$ for $n = 1, 2, 3, 4$. Parts are not allowed to move if $c_1 = c_2 = \infty$. (a_x, a_y) and $(c_{3,p}, c_{4,p})$ jointly decide the center of the part.

Details of our implementation of the deformation layer. Example 3, i.e. the formulation in (6), is used as our implementation of the deformation layer. With both spatial constraint and part scores considered, the summed map $\mathbf{B}$ is obtained using (6). The next step is to obtain the maximum score from the summed map $\mathbf{B}$, which corresponds to the formulation in (2). The same quadratic constraint is used for deformation layer model in (6) and the DPM model in (5). In order to obtain the maximum score from the summed map $\mathbf{B}$, the distance transform [23] used by DPM is used by our deformation layer for faster speed in the code provided online.

At the training stage, the gradient of a given loss L for the parameters $c_{n,p}$ are as follows:

$$\begin{aligned}
\frac{\partial L}{\partial c_{n,p}} &= \frac{\partial L}{\partial s_p}\frac{\partial s_p}{\partial c_{n,p}}\\
&= \frac{\partial L}{\partial s_p} d_n^{(\tilde{x}_p, \tilde{y}_p)},
\end{aligned} \tag{7}$$

where $(\tilde{x}_p, \tilde{y}_p) = \arg\max_{(x,y)} \mathbf{B}_p$, $\frac{\partial L}{\partial s_p}$ is the derivative propagated from the loss to the output s_p of the deformation layer. Therefore, only the value at location $(\tilde{x}_p, \tilde{y}_p)$ is used for learning the deformation parameter $c_{n,p}$ during the back-propagation.

3.5 Visibility reasoning and classification

The deformation layer in Section 3.4 provides the part scores $\mathbf{s} = \{s_1, \ldots, s_P\}$ using Eq. (2). $\mathbf{s}$ is then used for visibility reasoning and classification. We adopt the model in [46] to estimate visibility.

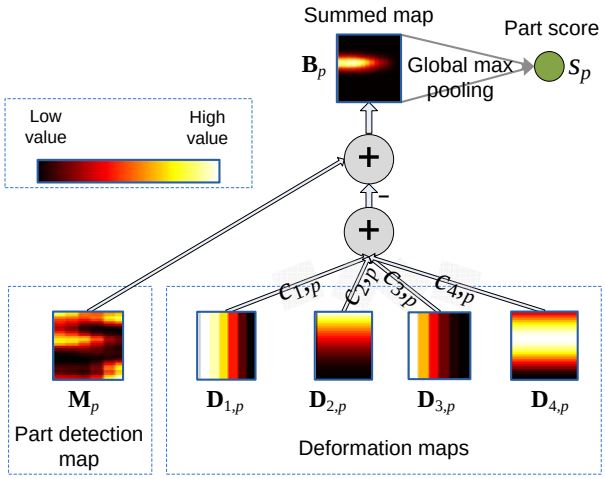


Figure 5. The deformation layer when deformation map is defined in (6). Part detection map $\mathbf{M}_p$ and deformation maps $\mathbf{D}_{n,p}$ are summed up with weights $c_{n,p}$ for $n = 1, 2, 3, 4$ to obtain the summed map $\mathbf{B}_p$. Global max pooling is then performed on the summed map $\mathbf{B}_p$ to obtain the score s_p for the pth part.

Fig. 6 shows the model for the visibility reasoning and classification in Fig. 2. Denote the score and visibility of the jth part at level l as s_j^l and h_j^l respectively. Denote the visibility of P_l parts at level l by $\mathbf{h}^l = [h_1^l \ \ldots h_{P_l}^l]^{\mathrm{T}}$. Given $\mathbf{s}$, the model for BP and inference is as follows:

$$\begin{aligned}
\tilde{h}_j^1 &= \sigma(c_j^1 + g_j^1 s_j^1),\\
\tilde{h}_j^{l+1} &= \sigma(\tilde{\mathbf{h}}^{l\mathrm{T}}\mathbf{w}_{*,j}^l + c_j^{l+1} + g_j^{l+1}s_j^{l+1}), l = 1, 2,\\
\tilde{y} &= \sigma(\tilde{\mathbf{h}}^{3\mathrm{T}}\mathbf{w}^{cls} + b),
\end{aligned} \tag{8}$$

where $\sigma(t) = (1 + exp(-t))^{-1}$ is the sigmoid function, g_j^l is the weight for s_j^l, c_j^l is its bias term, $\mathbf{W}^l$ models the correlation between $\mathbf{h}^l$ and $\mathbf{h}^{l+1}$, $\mathbf{w}_{*,j}^l$ is the jth column of $\mathbf{W}^l$, $\mathbf{w}^{cls}$ is considered as the linear classifier for the hidden units $\tilde{h}^3$, and $\tilde{y}$ is the estimated detection label. Hidden variables at adjacent levels are connected. $\mathbf{w}_{*,j}^l$ represents the relationship between $\tilde{\mathbf{h}}^l$ and $\tilde{h}_j^{l+1}$. A part can have multiple parents and multiple children. The visibility of one part is correlated with the visibility of other parts at the same level through shared parents. g_j^l, c_j^l, $\mathbf{W}^l$, $\mathbf{w}^{cls}$, and b are parameters to be learned.

The differences between the deep model in this paper and the approach in [46] are as follows:

1. The parts at levels 1 and 2 propagate information to the classifier through the parts at level 3 in [46]. But the imperfect part scores at level 3 may disturb the information from levels 1 and 2. This paper includes extra hidden nodes at levels 2 and 3. These nodes provide branches that help parts at level 1 and level 2 to directly propagate information to the classifier without being disturbed by other parts. These extra hidden nodes do not use detection scores and have the term $g_j^{l+1}s_j^{l+1} = 0$ in (8). They are represented by white circles in Fig. 6, while the hidden nodes with the term $g_j^{l+1}s_j^{l+1} \neq 0$ in (8) are represented by gray circles.

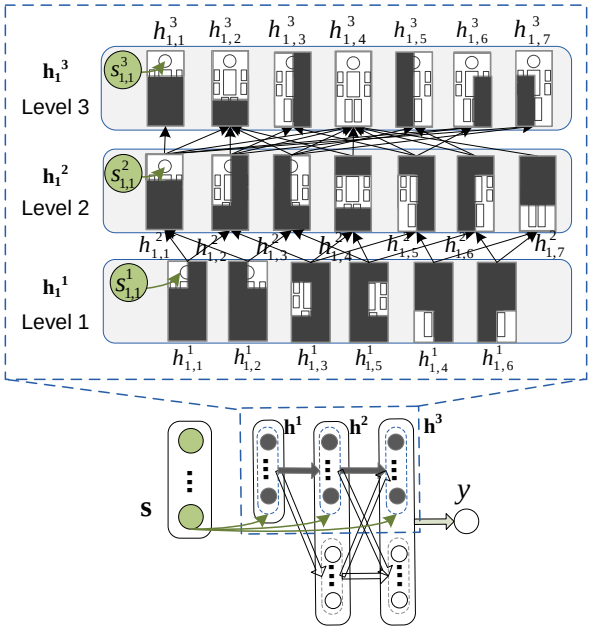


Figure 6. The visibility reasoning and detection label estimation model. For the ith part at the lth level, s_i^l is the detection score and h_i^l is the visibility. For example, h_1^1 indicates the visibility of the left-head-shoulder part. Best viewed in color.

2. The approach in [46] only learns the visibility relationship from part scores. Both HOG features and the parameters for the deformation model are fixed in [46]. In this paper, features, deformable models, and visibility relationships are jointly learned. In order to learn the parameters in the two convolutional layers and the deformation layer in Fig. 2, prediction error is back-propagated through $\mathbf{s}$. The gradient for $\mathbf{s}$ is:

$$\frac{\partial L}{\partial s_i^l} = \frac{\partial L}{\partial h_i^l}\frac{\partial h_i^l}{\partial s_i^l} = \frac{\partial L}{\partial h_i^l} h_i^l(1-h_i^l) g_i^l, \tag{9}$$

$$\text{where } \frac{\partial L}{\partial h_i^3} = \frac{\partial L}{\partial \tilde{y}} \tilde{y}(1-\tilde{y}) w_i^{cls},$$

$$\frac{\partial L}{\partial h_i^2} = w_{i,*}^2 \left[\frac{\partial L}{\partial \mathbf{h}^3} \odot \mathbf{h}^3 \odot (1-\mathbf{h}^3)\right], \tag{10}$$

$$\frac{\partial L}{\partial h_i^1} = w_{i,*}^1 \left[\frac{\partial L}{\partial \mathbf{h}^2} \odot \mathbf{h}^2 \odot (1-\mathbf{h}^2)\right],$$

$\odot$ denotes the Hadamard product; that is $(U \odot V)_{i,j} = U_{i,j}V_{i,j}$, $w_{i,*}^l$ is the ith row of $\mathbf{W}^l$, and w_i^{cls} is the ith element of the $\mathbf{w}^{cls}$. L is the loss function. For example $L = (y_{gnd} - \tilde{y})^2/2$ is for the square loss, and y_{gnd} the ground-truth label. $L = y_{gnd} \log \tilde{y} + (1 - y_{gnd}) \log(1 - \tilde{y})$ is for the log loss, which is chosen in this work.

In order to train this deep architecture, we adopt a multi-stage training strategy. We start with a 1-layer CNN using supervised training. Since Gabor filters are similar to the human visual system, they are used for initialing the first CNN. We add one more layer at each stage, the layers trained in the previous stage are used for initialization and then all the layers at the current stage are jointly optimized with BP.

4 The Extended Joint Deep Learning Model for Deep CNN

The deep model shown in Fig. 2 has only 2 convolutional layers. It is found that very deep models like VGG perform well on general object detection [26]. Therefore, we extend the model in Fig. 2 and propose an extended model for using the VGG and fast R-CNN in our framework. The extended model is shown in Fig. 7. In the extended framework, the following procedure is used in the deep model:

1) *The whole image* is treated as the input and then a deep CNN is used for extracting convolutional features. In our implementation, the 16-layer VGG [66] is used. The change from YUV in the basic model to RGB in the extended model is because the pretrained VGG model is based on the RGB input.
2) *The roi-pooled feature map* $pl5$ with fixed size (7×7) is obtained using the roi-pooling introduced in [25]. The input of the roi-pooling is the candidate region of interest (roi) and the last convolutional layer of VGG in our implementation. The rois are obtained using the region proposal network [61] in our implementation.
3) The roi-pooled feature map of size 7×7 in width and height is connected to two fully connected layers (fc6 and fc7). The features of fc7 is used for classification from the whole pedestrian region (fc8). The classification score from fc8 is treated as the detection score for the full body.
4) In order to obtain features with different sizes, we also connect the roi-pooled feature map to convolutional layers $\text{conv}6_1$, $\text{conv}6_2$, and $\text{conv}6_3$ with kernel sizes 5×5, 3×3, and 1×1 respectively. Padding is used for these convolutional layers so that their spatial sizes are unchanged after convolution, e.g. padding of 1 for kernel size 3×3.
5) Similar to the fully connected layers fc7 we use 1 convolutional layer $\text{conv}7_i$ to obtain features with more depth from $\text{conv}6_i$, $i = 1, 2, 3$.
6) Similar to fc8, we use 1 convolutional layers $\text{conv}8_i$ to obtain the part detection maps from the output of the layers $\text{conv}7_i$ for $i = 1, 2, 3$.
7) Part scores $\text{def}8_i$ are obtained from the part detection maps $\text{conv}8_i$ using the deformation layers.
8) Visibility reasoning and detection label estimation is conducted based on the scores fc8 and $\text{def}8_i$ for $i = 1, 2, 3$.

Fig. 8 shows the details on the kernel size, padding, output map size of the convolutional layers and deformation layers after the pooled features. The layers pl5, fc6, fc7 and fc8 are from the original VGG model. Other layers are added for obtaining scores of pedestrian parts with different sizes. The full-body part is used by both the basic model in Section 3 and the extended model. Pedestrian parts of different sizes are used in both the model in Section 3 and the extended model. Multi-resolution is not used in both models. Comparison between the original VGG layers and our additional layers is as follows:

1) Each neuron in the fc6 output map covers the whole pedestrian region. And the score from fc8 consider the whole pedestrian region. The fc6 layer from the original VGG can also be considered as a convolutional layer with kernel size 7×7 and without padding. The additional layers $\text{conv}6_1$, $\text{conv}6_2$, and $\text{conv}6_3$ cover partial pedestrian regions. A neuron in $\text{conv}6_3$ covers only a 1×1 roi-pooled region pl5 and a neuron in $\text{conv}6_3$ covers a 5×5 region. In this way, the part detection maps in $\text{conv}8_1$ are from the 5×5 region of the pl5. Similarly, the $\text{conv}8_2$ and

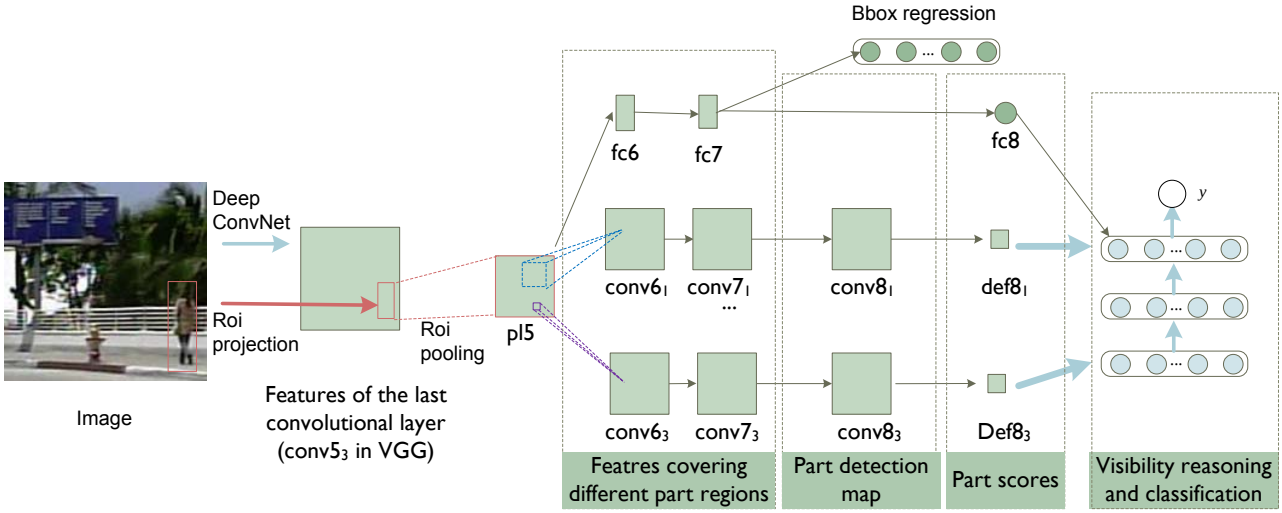


Figure 7. Joint learning of deformation and visibility using the VGG as baseline network and the fast R-CNN for obtaining features.

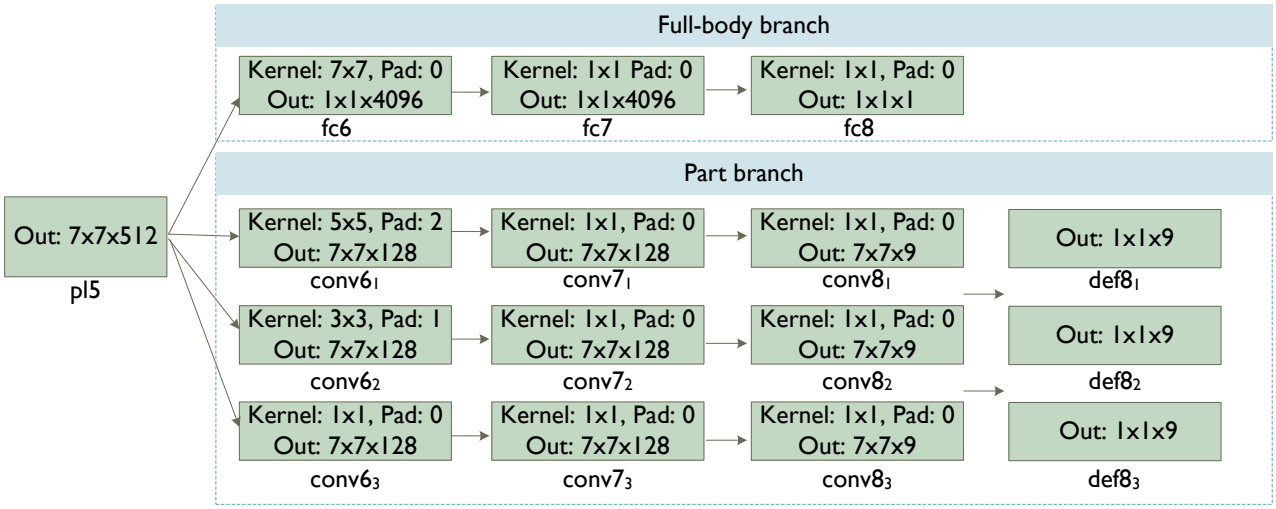


Figure 8. Details on the fully connected layers (fc6, fc7, fc8), convolutional layers (conv6$_i$, conv7$_i$, conv8$_i$ for $i = 1, 2, 3$) and deformation layers (def8$_1$, def8$_2$, def8$_3$) after the roi-pooling layer pl5. *Kernel* denotes the filter kernel size of convolution. *Pad* denotes the padding used for convolution. *Out* denotes the output map size. For example, 7×128 denotes the 128 maps of size 7×7 in width and height. There are 1 full-body score and 27 part scores obtained.

conv8$_3$ are, respectively, from the 3×3 and 1×1 region of pl5. Therefore, the part detection maps conv8$_1$, conv8$_2$ and conv8$_3$ are scores that consider pedestrian parts of different sizes but the same resolution. Fig. 15 shows the visualization of the filters learned for fc8, conv8$_1$, conv8$_2$, and conv8$_3$.

2) The fc7 layer is a fully connected layer. conv7$_i$ for $i = 1, 2, 3$ is a convolutional implementation of using fully connected layer for input feature maps of each spatial location separately. fc7 and conv7$_i$ do not change the receptive field of the features.
3) The output of layers fc6 and fc7 has 4096 channels. To save computation, the outputs of convolutional layers conv6$_i$ and conv7$_i$ have 128 channels.

4.1 Implementation details

The extended model is based on the Fast-RCNN framework. At the first stage, we use the VGG net for region proposal, which is pre-trained on ImageNet and finetuned on Caltech-Train. At the second stage, the extended deep model introduced in Section 4 is used for classifying the proposed regions as containing pedestrians or background. The second stage is called region classification stage.

Region proposal stage. We use the open source code released by Ren et al. [61] for obtaining candidate regions. We change some parameters of the region proposal network. For example, the sizes of anchors/boxes are $\{16\times16,\ 32\times32,\ 64\times64,\ 128\times128\}$, and the aspect ratios are set as $\{1.7,\ 2.4,\ 3.1\}$. During training and testing, we resize the image 640×480 to 1066×800 to fit the large input image size requirement of VGG. We use a learning rate 0.001 for the iterations [1 80000] and 0.0001 for the iterations [80000 120000]. After non-maximum suppression (NMS), about 2000 proposal regions per image are retained for training the extended deep model. In the testing stage, we keep boxes with RPN classification score larger than 0.2 as the proposed regions for the extended

deep model. With the threshold being 0.2, around 13 proposal regions per image are retained as the region proposal for pedestrian classification at the testing stage. More proposal regions are retained at the training stage for preserving more training samples, fewer proposal regions are retained at the testing stage for faster speed.

Region classification stage. The extended deep model for classification is shown in Fig. 7 and Fig. 8. It includes the full-body branch and the part branch. The roi-pooling layer proposed in [25] outputs feature map of size 7×7. With padding, the part score map is also 7×7. Denote (x, y) as horizontal location x and vertical location y in the 7×7 part score map, $x = \{0, 1, \ldots, 6\}, y = \{0, 1, \ldots, 6\}$. We select 9 anchor locations of parts, i.e. (2, 1), (2, 3), (2, 5), (3, 1), (3, 3), (3, 5), (4, 1), (4, 3) and (4, 5). The visibility reasoning layers are similar to UDN.

To train the extended deep model, we adopt two steps.

- *Step 1:* learn the extended deep model without visibility reasoning. Two kinds of implementations are tried in implementing the part branch. The first implementation places Relu before the deformation layer and uses $tanh$ as the activation function to get the deformable part score, and then uses the Euclidean loss for training part branch. The other implementation does not place non-linear activation function before the deformation layer and chooses sigmoid as the activation function to get the part score, and then uses the cross entropy loss for training the part branch. Both implementations have similar results.
- *Step 2:* with the model parameters in Step 1 as initial point, the extended deep model with visibility reasoning is trained.

In the training stage, layers from VGG are initialized by the VGG16 model pre-trained on ImageNet, and the other new layers are initialized from zero-mean Gaussian distribution.

The visibility reasoning layer is implemented by fully connected layer, they can be easily implemented by convolution. And the distance transform is used for the deformation layer, which is also applicable for convolution. The model in this paper can be made fully convolutional although we run it as a classifier for each window separately using the roi-pooling.

5 Experimental Results of the Basic Deep Model

The proposed framework is evaluated on the Caltech dataset [17] and the ETH dataset [20]. In order to save computation, a detector using HOG+CSS and Linear SVM is utilized for pruning candidate detection windows at both training and testing stages. Approximately 60,000 training samples that are not pruned by the detector are used for training the deep model. At the testing stage, the execution time required by our deep model is less than 10% of the execution time required by the HOG+CSS+SVM detector, which has filtered most samples. In the deep learning model, learning rate is fixed as 0.025 with batch size 60. Similar to [64], [33], norm penalty is not used.

The compared approaches are VJ [72], HOG [9], Shapelet [62], LatSVM-V1 [22], LatSVM-V2 [22], HogLbp [76], DBN-Isol [46], MultiFtr [77], MultiFtr+Motion [73], MultiResC [56], DBN-Mut [50], MultiSDP [87], LDCF [44], SCF+AlexNet [31], Katamari [4], SpatialPooling [55], SpatialPooling+ [55], SCCPriors [84], TA-CNN [69], CCF [82], CCF+CF [82], Checkerboards [91], DeepParts [68], CompACT-Deep [7], SDN [38], CrossTalk [14], HikSVM [40], FPDW [15], ACF [13], RandForest [41], and ConvNet-U-MS [64]. Existing approaches use various features, deformable part models and different learning approaches. The features used include Haar (VJ), HOG (HOG, LatSvm-V2), LBP (HogLbP), motion (MultiFtr+Motion) and geometric constraint (MultiResC). Different part models are used in LatSVM-V2, DBN-Isol and MultiResC. Different deep models are used by ConvNet-U-MS, DBN-Mut, SDN, CCF, MultiSDP, DeepParts, CompACT-Deep, SCF+AlexNet, and DN-HOG. The UDN is our basic deep model introduced in Section 3 and the UDN+ is our extended model in Section 4.

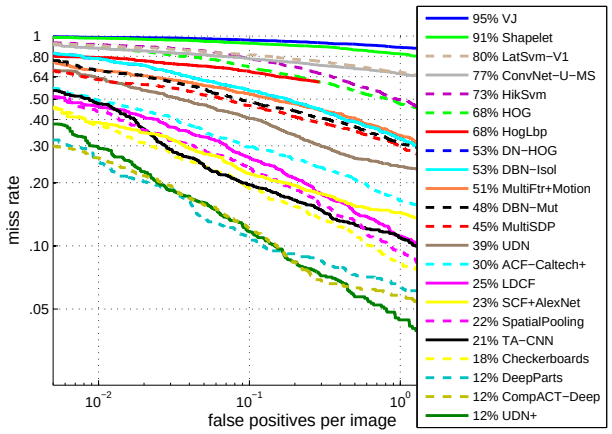


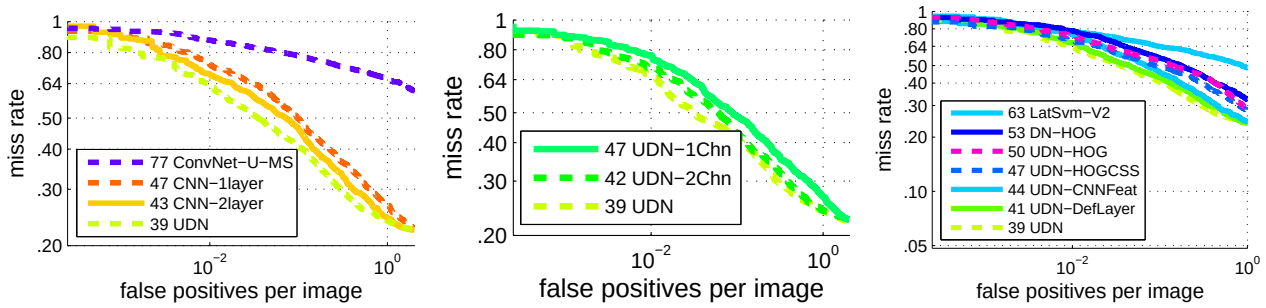
Figure 9. Overall results on the Caltech-Test dataset. The original annotation is used for training our UDN and UDN+ and evaluation on the test data. In the legend, 12% for our UDN+ model denotes the log average miss rate in $[10^{-2}, 10^{0}]$. Similarly for other approaches.

5.1 Results of the Caltech-Test dataset

The labels and evaluation code provided by Dollár *et al.* online are used for evaluation following the criteria proposed in [17]. As in [17], the *log-average miss rate* is used to summarize the detector performance, and is computed by averaging the miss rate at nine FPPI rates that are evenly spaced in the log-space in the range from 10^{-2} to 10^{0}. In the experiments, we evaluate the performance on the *reasonable* subset of the evaluated datasets. This subset, which is the most popular portion of the datasets, consists of pedestrians who are more than 49 pixels in height, and whose occluded portions are less than 35%.

To evaluate on the Caltech-Test dataset, the Caltech-Train dataset is used to train our model. The recent best performing approaches [12], [56] on Caltech-Test also use Caltech-Train as training data. At the training stage, there are approximately 60,000 negative samples and 4,000 positive samples from the Caltech-Train dataset.

Fig. 9 shows the overall experimental results on the Caltech-Test. Our UDN+ has better performance when compared with



(a) (b) (c)

Figure 10. Results of various designs of the deep model on the Caltech-Test dataset.

state-to-the-art methods. The basic UDN model has miss rate 39% and the extended model has miss rate 12%.

Since Caltech-Test is the largest among commonly used datasets, we investigate different designs of deep models on this dataset. Comparisons are shown in Figure 10.

Layer design. A one-layer CNN (CNN-1layer in Fig. 10(a)) is obtained by directly feeding the extracted features in Fig. 2 into a linear classifier. A two-layer CNN (CNN-2layer in Fig. 10(a)) is constructed by convolving the extracted feature maps with another convolutional layer and another pooling layer. Adding more convolutional and pooling layers on the top of the two-layer CNN does not improve the performance. Both CNNs have the same input and settings as the first convolutional layer and pooling layer of UDN, but do not have the deformation layer or the visibility estimation layer. This experiment shows that the usage of deformation and visibility layers outperforms CNNs. The ConvNet-U-MS in [64], which uses unsupervised feature learning for two-layer CNN, does not perform well on Caltech-Test. It has an average miss rate of 77%.

Input channel design. Fig. 10(b) shows the experimental results of investigating the influence of input channels introduced in Section 3.2. When the input data only has the first Y-channel image, the average miss rate is 47%. The inclusion of the second chennel of color images with a lower resolution reduces the miss rate by 5%. Including the third channel of edge maps reduces the miss rate by a further 3%.

Joint Learning. Fig. 10(c) shows the experimental results on investigating different degrees of joint learning. The first convolutional and pooling layers of UDN correspond to the feature extraction step. Therefore, the output of the two layers can be replaced by any other features, either manually designed or pre-learned.

- LatSvm-V2 [22], with a miss rate of 63%, manually designs the HOG feature, and then learns the deformation model. Visibility reasoning is not considered.
- DN-HOG [46], with a miss rate of 53%, fixes the HOG feature and the deformation model, and then learns the visibility model.
- UDN-HOG, with a miss rate of 50%, fixes the HOG feature, and then jointly learns the deformation and visibility layers with UDN. The difference between DN-HOG and UDN-HOG is whether deformation and visibility models are jointly learned.
- UDN-HOGCSS, with a miss rate of 47%, fixes the HOG+CSS feature, and jointly learns the deformation and visibility layers with UDN. Compared with UDN-HOG, the extra CSS feature reduces the miss rate by 3%.
- UDN-CNNFeat, with a miss rate of 44%, first learns the feature extraction layers using CNN-1layer in Fig. 10(a) and fixes these layers, and then jointly learns the deformation and visibility. In this case, the feature extraction is not jointly learned with the deformation and visibility. Compared with UDN-HOGCSS, UDN-CNNFeat reduces the miss rate by 3% by using the features learned from CNN-1layer.
- UDN-DefLayer, with a miss rate of 41%, jointly learns features and deformation. Visibility reasoning is not used.
- UDN jointly learns feature, deformation and visibility. Its miss rate is 5% lower than UDN-CNNFeat. Therefore, the interaction between deformation, visibility, and feature learning clearly improves the detection ability of the model.

5.2 Results of the ETH dataset

For a fair comparison on the ETH dataset, we follow the training setting commonly adopted by state-of-the-art approaches (including the best performing approaches [46], [22], [64] on ETH); that is, using the INRIA training dataset in [9] to train UDN. There are approximately $60,000$ negative samples and $2,000$ positive samples from the INRIA Training dataset, after the pruning of the HOG+CSS+SVM detector. Fig. 11 shows the experimental results on ETH. It can be seen that the basic UDN model [47] is still among the top ranking approaches on this dataset. Many studies (e.g., [33]) have found that deep models favor large-scale training data. The INRIA training set has fewer positive training samples than Caltech-Train. Therefore, the difference of miss rates between UDN and existing approaches is smaller than that on Caltech-Test.

Area under curve [64] is another measurement commonly used for evaluating the performance of pedestrian detection. Fig. 12 shows the average miss rate computed from AUC, which indicates that UDN also outperforms many sate-of-the-art methods under AUC.

6 Experimental Results for the Extended Deep Model on Caltech

In this section, we evaluate the proposed extended model introduced in Section 4 on the Caltech dataset. We evaluate

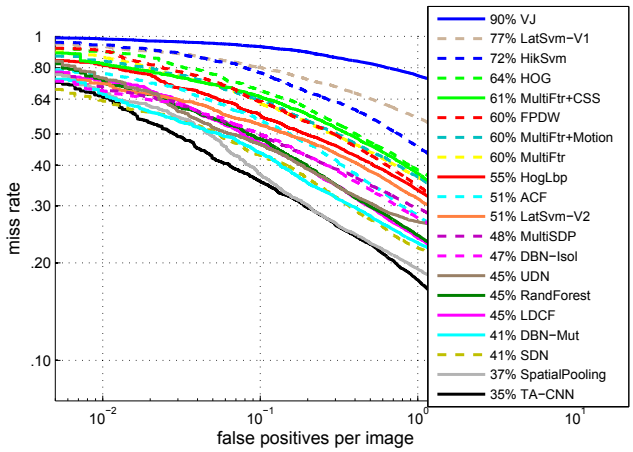


Figure 11. Experimental results on the ETH dataset.

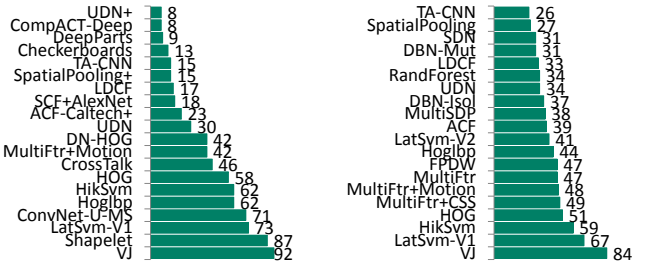


Figure 12. Comparisons of area under curve curve (AUC) on Caltech-Test (left) and ETH (right).

the performance on the *reasonable* subset of the evaluated datasets, which is for pedestrians over 50 pixels tall, with no or partial occlusion. In section 4 we use the training data of Caltech provided by Dollár et al. [17]. As pointed out by Zhang et al. in [90] , the label in [17] is noisy and influences both training and testing accuracy. The annotations are refined in [90]. In this section, we use the new annotations provided in [90] for both learning the model using the Caltch-Train and evaluation using the Caltech-Test.

After the region proposal step, the basic model takes about 0.1 seconds per image using Intel-i3 (4 CPUs) with 3.3GHz, and the extended model takes about 0.28 seconds per image using a single Titan X GPU. The region proposal network for region proposal takes about 0.2 seconds per image.

Fig. 13 shows the experimental comparison of the state of the art approaches and our final model. The compared approaches include RotatedFiltersNew10x VGG [90], CompACT-Deep [7], DeepParts [68], Checkerboards [91], TA-CNN [69], SCCPriors [84], SpatialPooling+ [55], SCF+AlexNet [31], Katamari [4], CCF [82], CCF+CF [82], LDCF [44], DBN-Isol [46], DBN-Mut [50], MultiSDP [87], HOG [9], and VJ [72]. The average miss rate is 10% in [90]. In these approaches, our result in Fig. 13 is based on single CNN model, while combination of hand crafted features and CNN features are used in RotatedFiltersNew10x VGG [90], CompACT-Deep [7], and DeepParts [68]. Fig. 13 provides log-average miss rate in $[10^{-2}, 10^{0}]$ and $[10^{-4}, 10^{0}]$. In the other experiments, we only provide the log-average miss rate in $[10^{-2}, 10^{0}]$.

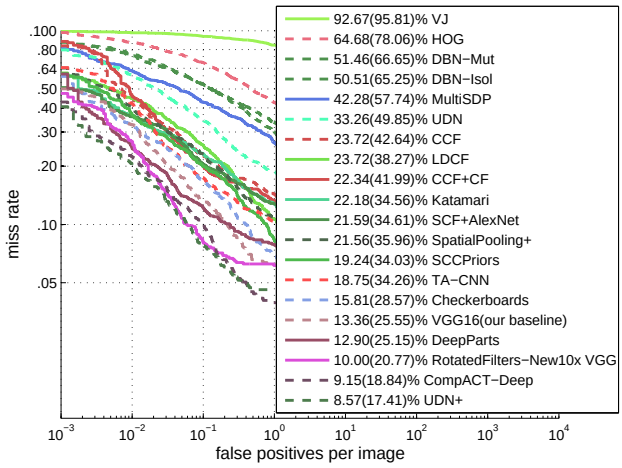


Figure 13. Comparison of our extended model with other state-of-the-art methods on the Caltech-Test dataset using the new labels provided in [90]. The labels in [90] are used for learning our UDN+ model. In the legend, 8.57% for our model denotes the log average miss rate in $[10^{-2}, 10^{0}]$, 17.41% for our model denotes the log average miss rate in $[10^{-4}, 10^{0}]$. Similarly for other approaches.

6.1 Experimental results on the components in the extended model

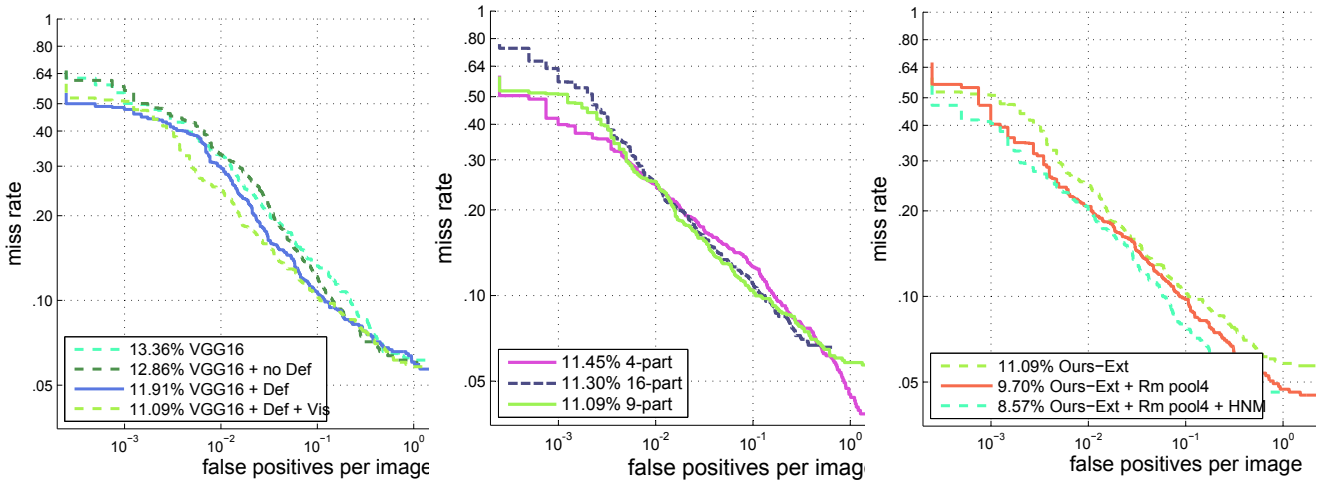
Fig. 14(a) shows the experimental results for deformation and visibility learning. The baseline VGG16 in Fig. 14(a) has average miss rate 13.36%. With the part branch, i.e. the *VGG16 + Def* in Fig. 14(a), the average miss rate is reduced to 11.91%. When the visibility reasoning layers are added, the model *VGG16 + Def + Vis* in Fig. 14(a) achieves 11.09% average miss rate.

We further examine the effectiveness of part branch and the deformation layer separately. The baseline VGG16 in Fig. 14(a) without the part branch has average miss rate 13.36%. The VGG with the part branch but without deformation layer, i.e. the *VGG16 + no Def* in Fig. 14(a), has average miss rate 12.86%. With deformation layer, the average miss rate is 11.91%, which is the *VGG16 + Def* in Fig. 14(a). The part branch without deformation reduces the absolute average miss rate by 0.5%, and the deformation layer further reduces the absolute average miss rate by 0.95%.

Our model without à *trous* has average miss rate 11.05% in Fig. 14(a). Based on the same setting, the model with à *trous* for increasing the resolution of feature has average miss rate 11.65%. The increase of resolution by à *trous* does not improve the detection accuracy for our model.

Based on the joint model, the experimental results investigating influence from the number of parts for each branch are shown in Fig. 14(b). It can be seen that the use of 9 parts is better than 4 or 16 parts. We use 9 parts as our final implementation. Since the model has 3 branches and each branch has 9 parts, there are 27 parts in all.

Higher resolution and hard negative mining for better accuracy. Based on VGG, our extended model has average miss rate 11.09%, which is Ours-Ext in Fig. 14(c). Based on this model, two commonly used approaches can be used for



(a) (b) (c)

Figure 14. Experimental results on the components in our framework. The baseline is the VGG16 model, *VGG16 + no Def* denotes the model with part branch without deformation layer. *VGG16 + Def* denotes the model with part branch and deformation layer. *VGG16 + Def + Vis* denotes the model with part branch, deformation layer and visibility reasoning. n-part in (b) denotes the *VGG16 + Def + Vis* model in (a) with n parts for each branch, $n = 4, 9, 16$. *Ours-Ext* in (c) corresponds to VGG16 + Def + Vis in (a) and 9-part in (b). *Ours-Ext + Rm pool4* denotes the model removing the max pooling at pool4 in VGG to increase the resolution of features. *Ours-Ext + Rm pool4 + HNM* denotes the model the use of hard negative mining on the model Ours-Ext + Rm pool4.

further improving accuracy. As shown in [17], many people are lower than 80 pixels. VGG16 stride is large for pedestrians with 50 pixels in height. Thus we remove pool4 for increasing the resolution of the feature maps in conv5_3. In this way, the average miss rate is reduced to 9.7%, which is the *Ours-Ext + Rm pool4* in Fig. 14(b). In order to remove common false positives, i.e. trees and poles, we fill each mini-batch with hard negatives in the training stage. The hard negatives can be obtained by using the detector on training set and selecting false positives with high score. On top of the the model *Ours-Ext + Rm pool4*, the use of hard negative mining, i.e. *Ours-Ext + Rm pool4 + HNM* as shown in Fig. 14(b), has average miss rate 8.57%.

6.2 Visualization of the learned model parameters

Fig. 15 shows the visualization of the learned filters for full body and body parts using the DeepVisualization in [1]. The visualized filter for the full-body branch in Fig. 15(a) covers the full body of a pedestrian. It is not good for representing the legs of the pedestrian. The filters for the part branch in Fig. 15 (b)-(d) are stitched for better visualization quality. It can be seen that the filters for parts provide more details of pedestrians. Fig. 16 shows the learned deformation map.

Several detection results of DeepParts [68], CompACT-Deep [7] and our framework are shown in Fig. 17(a). Compared with other approaches, our approach has less false positives and has more accurate bounding box in localizing the pedestrian, especially in the regions of crowd. We can use fewer detection boxes than others to achieve the same miss rate.

We also show some false positives and missing pedestrians of our framework in Fig. 17 (b). Many false positives appear on the cars. These can be removed with the labels of cars, contextual information, and more negatives samples of cars. Some missing pedestrians have high occlusion or low resolution.

7 Conclusion

This paper proposes a unified deep model that jointly learns four components – feature extraction, deformation handling, occlusion handling and classification – for pedestrian detection. Through interaction among these interdependent components, joint learning achieves detection accuracy improvement on benchmark pedestrian detection datasets. Detailed experimental comparisons clearly show that the proposed new model can maximize the strength of each component when all the components cooperate with each other. We enrich the deep model by introducing the deformation layer, which has great flexibility to incorporate various deformation handling approaches. We expect even larger improvement by training our UDN on much larger-scale training sets in the future work.

Acknowledgment: This work is supported by the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project No. CUHK 417110, CUHK 417011, CUHK 429412, CUHK 1420611, CUHK 14206114, CUHK 14205615) and National Natural Science Foundation of China (Project No. 61005057).

References

[1] Deepvisualization. DeepVisualization on https://github.com/happynear/DeepVisualization 11, 12
[2] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *ECCV*, 2010. 2
[3] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using hough transforms. In *CVPR*, 2010. 1, 2
[4] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV Workshop*, pages 613–627. Springer, 2014. 2, 8, 10

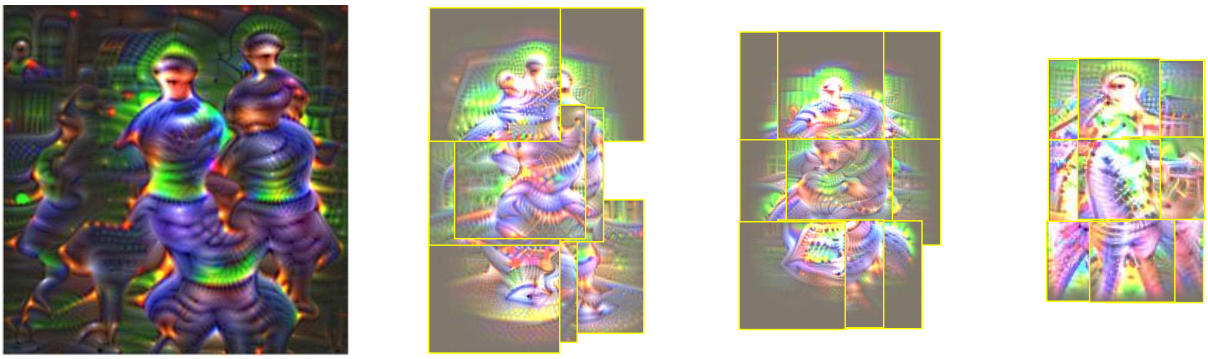


Figure 15. The filters for full-body branch (a) and part branch (b)(c)(d) visualized using the code provided on [1]. The body parts in (b)(c)(d) are visualized by stitching them at different anchor locations. More details can be found in Section 4.

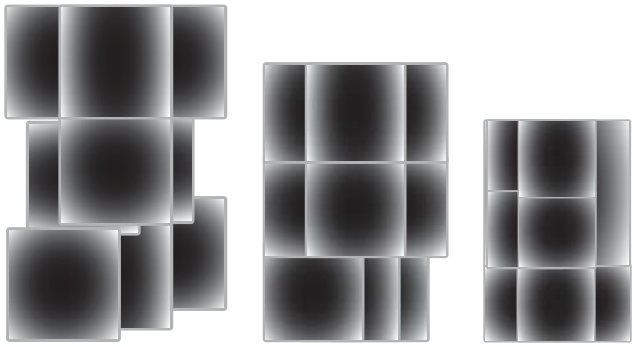


Figure 16. Visualization of the learned defomation layer parameters for large parts (left), median-sized parts (middle) and small parts (right).

[5] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. 2

[6] L. Bourdev and J. Malik. Poselets: body part detectors trained using 3D human pose annotations. In *ICCV*, 2009. 2

[7] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*, 2015. 8, 10, 11

[8] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*, 2014. 2

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2, 8, 9, 10

[10] C. Desai and D. Ramanan. Detecting actions, poses, and objects with relational phraselets. In *ECCV*, 2012. 2

[11] M. Dikmen, D. Hoiem, and T. S. Huang. A data-driven method for feature transformation. In *CVPR*, 2012. 2

[12] Y. Ding and J. Xiao. Contextual boost for pedestrian detection. In *CVPR*, 2012. 1, 8

[13] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Trans. PAMI*, 36(8):1532–1545, 2014. 2, 8

[14] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*, 2012. 1, 2, 8

[15] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, 2010. 8

[16] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. 1, 2

[17] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: an evaluation of the state of the art. *IEEE Trans. PAMI*, 34(4):743 – 761, 2012. 1, 2, 8, 10, 11

[18] M. Enzweiler, A. Eigenstetter, B. Schiele, and D. M. Gavrila. Multi-cue pedestrian classification with partial occlusion handling. In *CVPR*, 2010. 1, 2

[19] D. Erhan, Y. Bengio, A.Courville, and P. Vincent. Visualizing higher-layer features of deep networks. Technical report, University of Montreal, 2009. 4

[20] A. Ess, B. Leibe, and L. V. Gool. Depth and appearance for mobile scene analysis. In *ICCV*, 2007. 2, 8

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. PAMI*, 30:1915–1929, 2013. 2

[22] P. Felzenszwalb, R. B. Grishick, D.McAllister, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 32:1627–1645, 2010. 1, 2, 4, 5, 8, 9

[23] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005. 2, 5

[24] T. Gao, B. Packer, and D. Koller. A segmentation-aware object detection model with occlusion handling. In *CVPR*, 2011. 1

[25] R. Girshick. Fast r-cnn. In *CVPR*, 2015. 6, 8

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 6

[27] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015. 2

[28] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 2

[29] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, July 2006. 2

[30] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006. 2

[31] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, pages 4073–4082, 2015. 2, 8, 10

[32] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *CVPR*, 2009. 2

[33] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 3, 8, 9

[34] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 2, 4

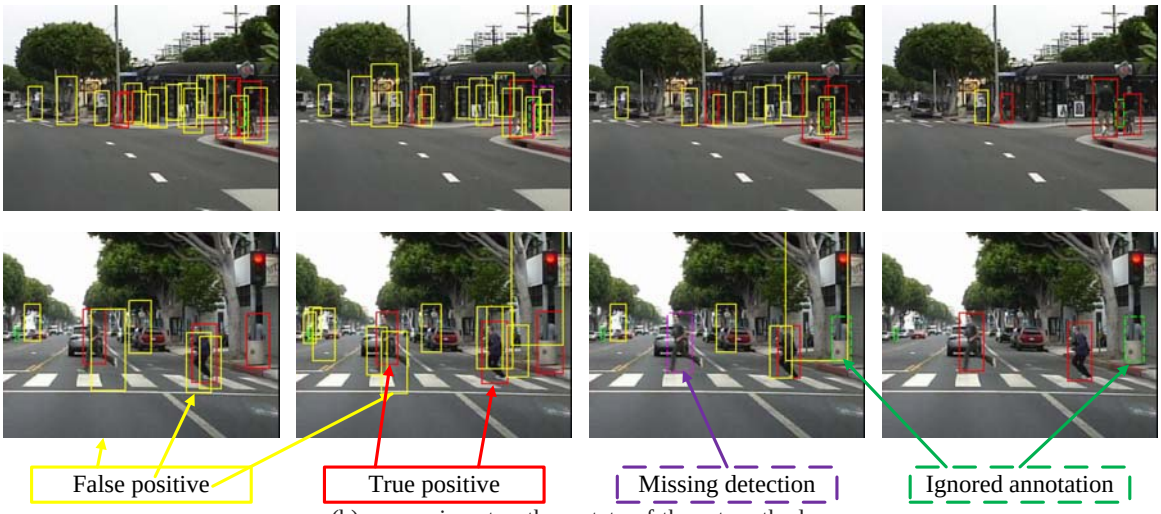
[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2, 3

[36] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR*, 2005. 2

[37] D. Lowe. Distinctive image features from scale-invarian keypoints. *IJCV*, 60(2):91–110, 2004. 1

[38] P. Luo, Y. Tian, X. Wang, and X. Tang. Switchable deep network for pedestrian detection. In *CVPR*, 2014. 2, 8

(a) some missing pedestrians (red boxes) and high-score false positives (yellow boxes).



(b) comparison to others state-of-the-art methods.

Figure 17. Some missing pedestrians and high-score false positives (a) and the detection results compared with other state-of-the-art methods (b). In (b), the columns from left to right are TA-CNN, DeepParts, CompACT-Deep and ours.

[39] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *CVPR*, 2012. 2

[40] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008. 1, 2, 8

[41] J. Marın, D. Vázquez, A. M. López, J. Amores, and B. Leibe. Random forests of local experts for pedestrian detection. In *CVPR*, 2013. 8

[42] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool. Handling occlusions with franken-classifiers. In *ICCV*, 2013. 1, 2

[43] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, 2006. 2

[44] W. Nam, P. Dollár, and J. H. Han. Local decorrelation for improved pedestrian detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2014. 8, 10

[45] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *CVPR*, 2009. 2

[46] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *CVPR*, 2012. 1, 2, 5, 6, 8, 9, 10

[47] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *ICCV*, 2013. 2, 9

[48] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. In *CVPR*, 2013. 2

[49] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015. 1, 2

[50] W. Ouyang, X. Zeng, and X. Wang. Modeling mutual visibility relationship in pedestrian detection. In *CVPR*, 2013. 1, 8, 10

[51] W. Ouyang, X. Zeng, and X. Wang. Single-pedestrian detection aided by two-pedestrian detection. *IEEE Trans. PAMI*, 37(9):1875–1889, Sept. 2015. 2

[52] W. Ouyang, X. Zeng, and X. Wang. Learning mutual visibility relationship for pedestrian detection with a deep model. *IJCV*, 120(1):14–27, 2016. 2

[53] W. Ouyang, X. Zeng, and X. Wang. Partial occlusion handling in pedestrian detection with a deep model. *IEEE Trans. Circuits Syst. Video Technol.*, 26(11):2123–2137, Nov 2016. 2

[54] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, K. Wang, J. Yan, C.-C. Loy, and X. Tang. Deepid-net: Deformable deep convolutional neural networks for object detection. *IEEE Trans. PAMI*, page accepted, 2016. 1

[55] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Pedestrian detection with spatially pooled features and structured ensemble learning. *IEEE Trans. PAMI*, 38(6):1243–1257, 2016. 8, 10

[56] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *ECCV*, 2010. 2, 8

[57] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring weak stabilization for motion feature extraction. In *CVPR*, 2013. 2

[58] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *UAI*, 2011. 2

[59] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2007. 4

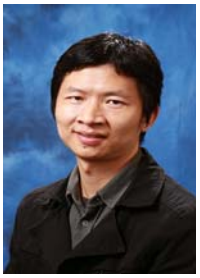
[60] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. Lecun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 2

[61] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 6, 7

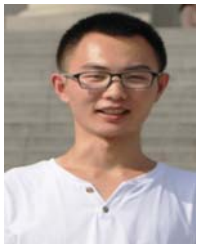
[62] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *CVPR*, 2007. 8

[63] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *ICCV*, 2009. 2

[64] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun. Pedestrian detection with unsupervised and multi-stage feature learning. In *CVPR*, 2013. 2, 8, 9
[65] V. D. Shet, J. Neumann, V. Ramesh, and L. S. Davis. Bilattice-based logical reasoning for human detection. In *CVPR*, 2007. 2
[66] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 6
[67] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for computing face similarities. In *ICCV*, 2013. 2
[68] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep learning strong parts for pedestrian detection. In *ICCV*, 2015. 2, 8, 10, 11
[69] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *CVPR*, 2015. 2, 8, 10
[70] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Trans. PAMI*, 30(10):1713–1727, Oct. 2008. 1, 2
[71] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. 2
[72] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2):153–161, 2005. 1, 2, 8, 10
[73] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR*, 2010. 2, 8
[74] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 2
[75] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016. 2
[76] X. Wang, X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *CVPR*, 2009. 1, 2, 8
[77] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *DAGM*, 2008. 8
[78] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *ICCV*, 2005. 2
[79] T. Wu and S. Zhu. A numeric study of the bottom-up and top-down inference processes in and-or graphs. *IJCV*, 93(2):226–252, Jun. 2011. 2
[80] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In *CVPR*, 2014. 2
[81] J. Yan, Y. Yu, X. Zhu, Z. Lei, and S. Z. Li. Object detection by labeling superpixels. In *CVPR*, 2015. 1, 2
[82] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features. In *ICCV*, 2015. 1, 2, 8, 10
[83] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 2
[84] Y. Yang, Z. Wang, and F. Wu. Exploring prior knowledge for pedestrian detection. In *BMVC*, 2015. 8, 10
[85] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013. 2
[86] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 2
[87] X. Zeng, W. Ouyang, and X. Wang. Multi-stage contextual deep learning for pedestrian detection. In *ICCV*, 2013. 2, 8, 10
[88] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang. Gated bi-directional cnn for object detection. In *ECCV*, 2016. 1
[89] S. Zhang, C. Bauckhage, and A. Cremers. Informed haar-like features improve pedestrian detection. In *CVPR*, pages 947–954, 2014. 2
[90] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele. How far are we from solving pedestrian detection? *arXiv preprint arXiv:1602.01237*, 2016. 10
[91] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *CVPR*, 2015. 2, 8, 10
[92] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015. 2
[93] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010. 1, 2



Wanli Ouyang received the PhD degree in the Department of Electronic Engineering, The Chinese University of Hong Kong, where he is now a Research Assistant Professor. His research interests include image processing, computer vision and pattern recognition.



Hui Zhou received the bachelor degree at university of science and electronic technology of china in 2015. He is currently a Research Assistant in the Department of Electronic Engineering at The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Xiaogang Wang received the PhD degree from the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology in 2009. He is currently an Associate Professor in the Department of Electronic Engineering at The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Hongsheng Li received the masters and doctorate degrees in computer science from Lehigh University, Pennsylvania, in 2010 and 2012, respectively. He is an associate professor in the School of Electronic Engineering at University of Electronic Science and Technology of China. His research interests include computer vision, medical image analysis, and machine learning.

Junjie Yan received the PhD degree in 2015 from National Laboratory of Pattern Recognition, Chinese Academy of Sciences. His research focus on object detection, face analysis, and deep learning. Since 2016, he is a principal engineer at SenseTime Group Limited and leads the R&D in video surveillance.

Quanquan Li received his bachelor degree in information engineering from The Chinese University of Hong Kong in 2015. He joined SenseTime as a researcher in 2015. He has worked on object detection, particularly on pedestrian detection and car detection under surveillance scenes.