Joint Deep Learning for Pedestrian Detection

Wanli Ouyang and Xiaogang Wang
Department of Electronic Engineering, the Chinese University of Hong Kong
wlouyang, xgwang@ee.cuhk.edu.hk

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[1]*. By establishing automatic, mutual interaction among components, the deep model achieves a* 9% *reduction in the average miss rate compared with the current best-performing pedestrian detection approaches on the largest Caltech benchmark dataset.*

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 [2, 5, 12, 47, 8]. 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 interdependent components are important. First, features should capture the most discriminative information of pedestrians. Well-known features such as Haar-like features [49], SIFT [29], and HOG [5] are designed to be robust to intra-class variation while remain sensitive to inter-class variation. 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 [17] allows human parts to articulate with constraint. Third, occlusion handling approaches [13, 51, 19] 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

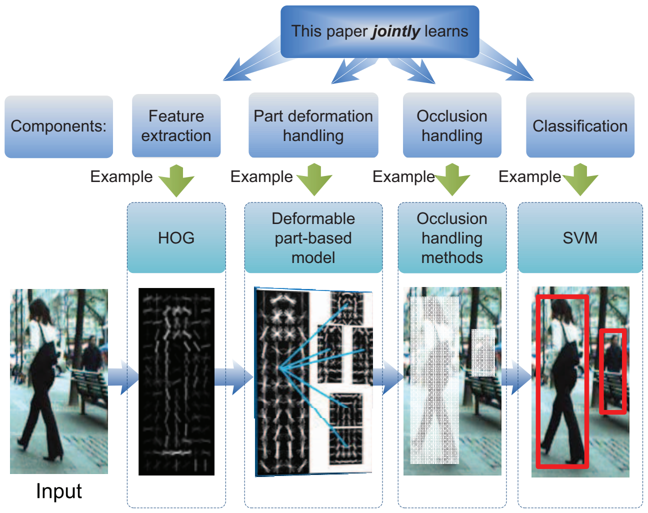

This paper *jointly* learns
Components:
Feature extraction
Part deformation handling
Occlusion handling
Classification
Example
Example
Example
Example
HOG
Deformable part-based model
Occlusion handling methods
SVM
Input

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.

enclosing a pedestrian. SVM [5], boosted classifiers [11], random forests [9], 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 [5]. Then HOG feature become fixed when people design new classifiers [31]. (2) A few HOG feature parameters are tuned in [17] and fixed, and then different part models are learned in [17, 58]. (3) By fixing HOG features and deformable models, occlusion handling models are learned in [34, 36], 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

[1] Code available www.ee.cuhk.edu.hk/~wlouyang/projects/ouyangWiccv13Joint/index.html

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.
2. We enrich the operation in deep models by incorporating the deformation layer into the convolutional neural networks (CNN) [26]. 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 [3]. They have achieved spectacular progress in computer vision [20, 21, 40, 23, 25, 33, 24, 56, 30, 46, 16, 38]. Deep models for pedestrian detection focus on feature learning [44, 33], contextual information learning [57], and occlusion handling [34].

Many features are utilized for pedestrian detection. Haar-like features [49], HOG [5], and dense SIFT [48] are designed to capture the overall shape of pedestrians. First-order color features like color histograms [11], second-order color features like color-self-similarity (CSS) [50] and co-occurrence features [43] are also used for pedestrian detection. Texture feature like LBP are used in [51]. Other types of features include the covariance descriptor [47], depth [15], segmentation results [13], 3D geometry [22], and their combinations [27, 51, 11, 50, 13, 43]. All the features mentioned above are designed manually. Recently, researchers have become aware of the benefit of learning features from training data [1, 33, 44]. 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 approach in [7] learns 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 [17, 58, 37, 35] for handling translational movement of parts. To handle more complex articulations, size change and rotation of parts are modeled in [18], and mixture of part appearance and articulation types are modeled in [4, 55, 6]. In these approaches, features are manually designed.

In order to handle occlusion, many approaches have been proposed for estimating the visibility of parts [13, 51, 54, 53, 45, 27]. Some of them use the detection scores of blocks or parts [51, 34, 13, 54] as input for visibility estimation. Some use other cues like segmentation results [27, 13] and depth [13]. However, all these approaches learn the occlusion modeling separately from feature extraction and part models.

The widely used classification approaches include various boosting classifiers [9, 11, 53], linear SVM [5], histogram intersection kernel SVM [31], latent SVM [17], multiple kernel SVM [48], structural SVM [58], and probabilistic models [2, 32]. 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. Method

3.1. Overview of the proposed deep model

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

1. *Filtered data maps* are obtained from the first convolutional layer. This layer convolves the 3-channel input image data with $9 \times 9 \times 3$ filters and outputs 64 maps. $|tanh(x)|$, i.e. activation function $tanh$ and absolution value rectification, is used for each filter response x.
2. *Features maps* are obtained by average pooling of the 64 filtered data maps 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 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 handling 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).

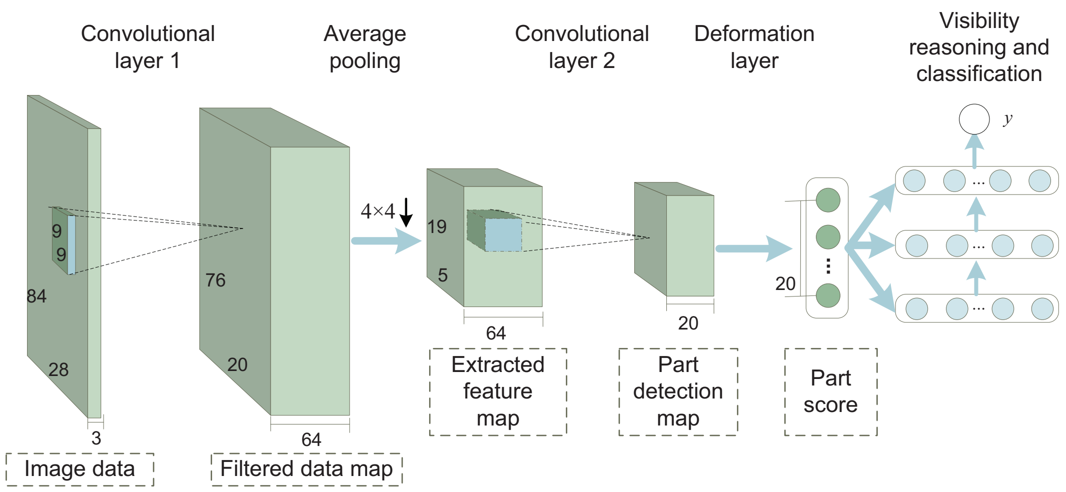


Figure 2. Overview of our deep model. Image data is convolved with 64 $9 \times 9 \times 3$ filters and averagely pooled to obtain 64 feature maps. The feature maps are then processed by the second convolutional layer and the deformation layer to obtain 20 part scores. Finally the visibility reasoning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

3.2. Input data preparation

The detection windows are extracted into images with height 84 and width 28, in which pedestrians have height 60 and width 20. The input image data contains three channels.

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

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

(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 images 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.

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 [26, 24]. Since the parts of pedestrians have different sizes, we design the filters in the second convolutional layer with variable sizes. As shown in Fig. 3(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. 3(a). 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

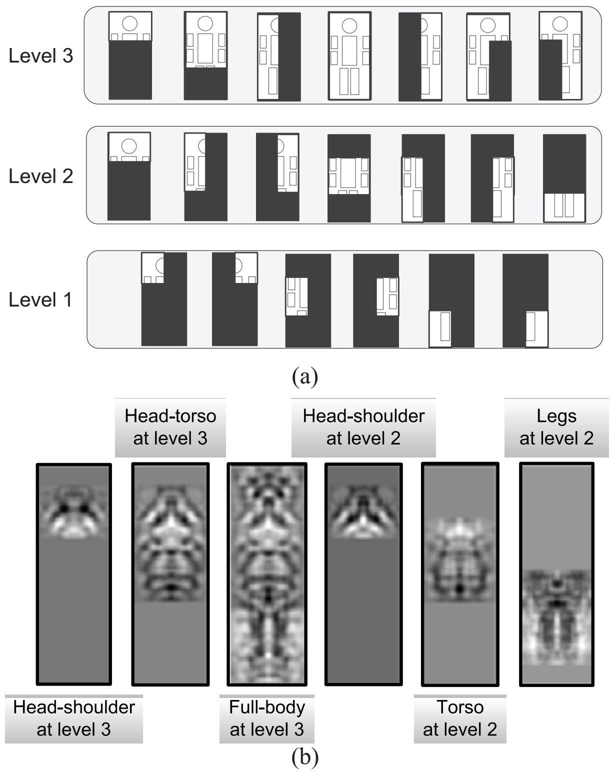


Figure 3. The parts model (a) and the filters (b) learned at the second convolutional layer. We follow [14] and visualize the filter that optimizes the corresponding stimuli of the neurons, which is also used in [25].

middle level respectively) because this body part itself can generate an occlusion status. Fig. 3(b) shows a few part filters learned with our deep model. They are visualized using the activation maximization approach in [14]. 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. 3(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. 3(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.

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 detection maps separately and produces the pth part score s_p from the pth part detection map, denoted by $\mathbf{M}_p$. 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:

$$\mathbf{B}_p = \mathbf{M}_p + \sum_{n=1}^{N} c_{n,p}\mathbf{D}_{n,p}. \tag{1}$$

$\mathbf{D}_{n,p}$ 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):

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

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:

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

At the training stage, only the value at location $(x, y)_p$ of $\mathbf{B}_p$ is used for learning the deformation parameters.

The $c_{n,p}$ and $\mathbf{D}_{n,p}$ in (1) are the key 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.

Example 1. Suppose $N = 1$, $c_{1,p} = 1$ and the deformation map $\mathbf{D}_{1,p}$ is 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). The approach in [39] treats deformation as bins of locations.

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 does not adapt to different deformation properties of different parts.

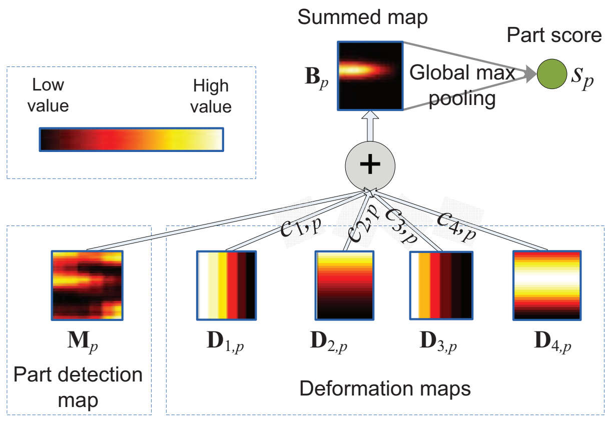


Figure 4. The deformation layer when deformation map is defined in (4). Part detection map and deformation maps are summed up with weights $c_{n,p}$ for $n = 1, 2, 3, 4$ to obtain the summed map $\mathbf{B}_p$. Global max pooling is then performed on the summed map to obtain the score s_p for the pth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

Example 3. The deformation layer can represent the widely used quadratic constraint of deformation in [17]. Below, we skip the subscript $_p$ used in Eq. (1) to be concise. The quadratic constraint of deformation can be represented as follows:

$$b^{(x,y)} = m^{(x,y)} + c_1(x - a_x + \frac{c_3}{2c_1})^2 + c_2(y - a_y + \frac{c_4}{2c_2})^2, \tag{4}$$

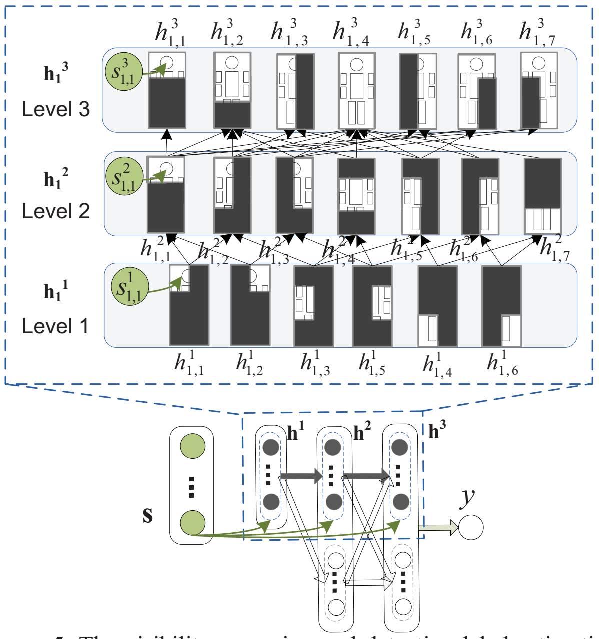
where $m^{(x,y)}$ is the (x, y)th element of the part detection map $\mathbf{M}$, (a_x, a_y) is the predefined anchor location of the pth part. They are adjusted by $c_3/2c_1$ and $c_4/2c_2$, which are automatically learned. c_1 and c_2 (4) decide the deformation cost. There is no deformation cost if $c_1 = c_2 = 0$. Parts are not allowed to move if $c_1 = c_2 = -\infty$. (a_x, a_y) and $(\frac{c_3}{2c_1}, \frac{c_4}{2c_2})$ jointly decide the center of the part. The quadratic constraint in Eq. (4) can be represented using Eq. (1) as follows:

$$\begin{aligned}
&\mathbf{B} = \mathbf{M} + c_1\mathbf{D}_1 + c_2\mathbf{D}_2 + c_3\mathbf{D}_3 + c_4\mathbf{D}_4 + c_5 \cdot \mathbf{1},\\
&b^{(x,y)} = m^{(x,y)} + c_1 d_1^{(x,y)} + c_2 d_2^{(x,y)} + c_3 d_3^{(x,y)} + c_4 d_4^{(x,y)} + c_5,\\
&d_1^{(x,y)} = (x - a_x)^2, \;\; d_2^{(x,y)} = (y - a_y)^2, d_3^{(x,y)} = x - a_x,\\
&d_4^{(x,y)} = y - a_y, c_5 = {c_3}^2/(4c_1) + {c_4}^2/(4c_2),
\end{aligned} \tag{5}$$

where $\mathbf{1}$ is a matrix with all elements being one, $d_n^{(x,y)}$ is the (x, y)th element of $\mathbf{D}_n$. In this case, c_1, c_2, c_3 and c_4 are parameters to be learned and $\mathbf{D}_n$ are predefined. c_5 is the same in all locations and need not be learned. Fig. 4 illustrates this example, which is used as the deformation layer in this work.

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 rea-



$h^3_{1,1}$ $h^3_{1,2}$ $h^3_{1,3}$ $h^3_{1,4}$ $h^3_{1,5}$ $h^3_{1,6}$ $h^3_{1,7}$
$\mathbf{h}_1{}^3$
Level 3
$s^3_{1,1}$
$\mathbf{h}_1{}^2$
Level 2
$s^2_{1,1}$
$h^2_{1,1}$ $h^2_{1,2}$ $h^2_{1,3}$ $h^2_{1,4}$ $h^2_{1,5}$ $h^2_{1,6}$ $h^2_{1,7}$
$\mathbf{h}_1{}^1$
Level 1
$s^1_{1,1}$
$h^1_{1,1}$ $h^1_{1,2}$ $h^1_{1,3}$ $h^1_{1,5}$ $h^1_{1,4}$ $h^1_{1,6}$
$\mathbf{s}$
$\mathbf{h}^1$ $\mathbf{h}^2$ $\mathbf{h}^3$
y

Figure 5. 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.

soning and classification. We adopt the model in [34] to estimate visibility.

Fig. 5 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 $\mathbf{h}^l = [h_1^l \ \dots h_{P_l}^l]^{\mathrm{T}}$. Given $\mathbf{s}$, the model for BP and inference is as follows:

$$\tilde{h}_j^1 = \sigma(c_j^1 + g_j^1 s_j^1),$$
$$\tilde{h}_j^{l+1} = \sigma(\tilde{\mathbf{h}}^{l\mathrm{T}}\mathbf{w}_{*,j}^l + c_j^{l+1} + g_j^{l+1}s_j^{l+1}), l = 1, 2, \tag{6}$$
$$\tilde{y} = \sigma(\tilde{\mathbf{h}}^{3\mathrm{T}}\mathbf{w}^{cls} + b),$$

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 [34] are as follows:

1. The parts at levels 1 and 2 propagate information to the classifier through the parts at level 3 in [34]. 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 (6). They are represented by white circles in Fig. 5, while the hidden nodes with the term $g_j^{l+1}s_j^{l+1} \neq 0$ in (6) are represented by gray circles.

2. The approach in [34] only learns the visibility relationship from part scores. Both HOG features and the parameters for the deformation model are fixed in [34]. 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:

$$\frac{\partial L}{\partial s_i^l} = \frac{\partial L}{\partial h_i^l}\frac{\partial h_i^l}{\partial s_i^l} = \frac{\partial L}{\partial h_i^l} h_i^l(1-h_i^l)g_i^l, \tag{7}$$

$$\text{where } \frac{\partial L}{\partial h_i^3} = \frac{\partial L}{\partial \tilde{y}}\tilde{y}(1-\tilde{y})w_i^{cls},$$
$$\frac{\partial L}{\partial h_i^2} = w_{i,*}^2\left[\frac{\partial L}{\partial \mathbf{h}^3} \odot \mathbf{h}^3 \odot (1-\mathbf{h}^3)\right], \tag{8}$$
$$\frac{\partial L}{\partial h_i^1} = w_{i,*}^1\left[\frac{\partial L}{\partial \mathbf{h}^2} \odot \mathbf{h}^2 \odot (1-\mathbf{h}^2)\right],$$

$\odot$ 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. Experimental Results

The proposed framework is evaluated on the Caltech dataset [12] and the ETH dataset [15]. 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 [44, 24], norm penalty is not used.

The labels and evaluation code provided by Dollár *et al.* online are used for evaluation following the criteria proposed in [12]. As in [12], 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%.

The compared approaches are VJ [49], Shapelet [42], PoseInv [28], LatSVM-V1 [17], LatSVM-V2 [17], HikSVM [31], HOG [5], MultiFtr [52], HogLbp [51], Pls [43], MultiFtr+CCS, MultiFtr+Motion [50], FeatSynth [1] FPDW [10], ChnFtrs [11], MultiResC [37], CrossTalk [9], DN-HOG [34] and ConvNet-U-MS

[44]. Existing approaches use various features, deformable part models and different learning approaches. The features used include Haar (VJ), HOG (HOG, LatSvm-V2), CSS (MultiFtr+CCS), LBP (HogLbP), motion (MultiFtr+Motion) and geometric constraint (MultiResC). Different part models are used in LatSVM-V2, DN-HOG and MultiResC. Different deep models are used by ConvNet-U-MS and DN-HOG. Our unified deep net is denoted by UDN.

4.1. Results of the Caltech-Test dataset

To evaluate on the Caltech-Test dataset, the Caltech-Train dataset is used to train our model. The recent best performing approaches [8, 37] 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. 6 shows the overall experimental results on the Caltech-Test. The current best performing approaches on the Caltech-Test are the MultiResC [37] and the contextual boost [8], both of which have an average miss rate of 48%. Our approach reduces the average miss rate by 9%.

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 7.

Layer design. A one-layer CNN (CNN-1layer in Fig. 7(a)) is obtained by directly feeding the extracted features in Fig. 2 into a linear classifier. A two-layer CNN (CNN-2layer in Fig. 7(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 [44], 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. 7(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. 7(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 [17], with a miss rate of 63%, manually designs the HOG feature, and then learns the deformation model. Visibility reasoning is not considered.
- DN-HOG [34], 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. 7(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. Visiblity 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.

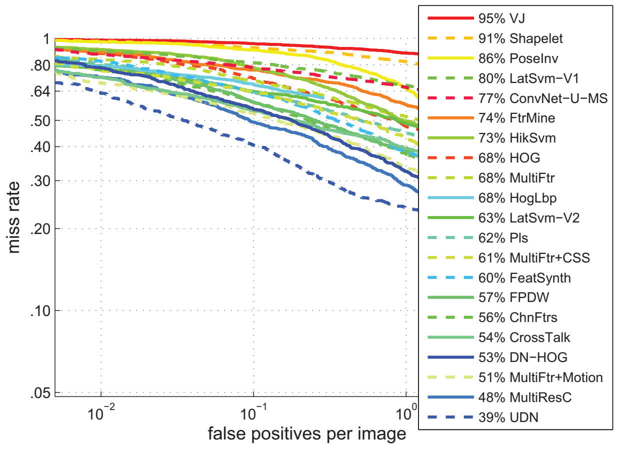


Figure 6. Overall results on the Caltech-Test dataset.

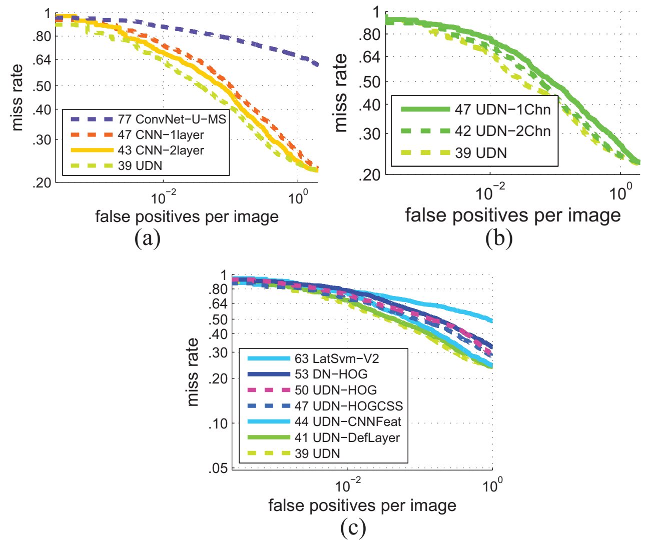


Figure 7. Results of various designs of the deep model on the Caltech-Test dataset.

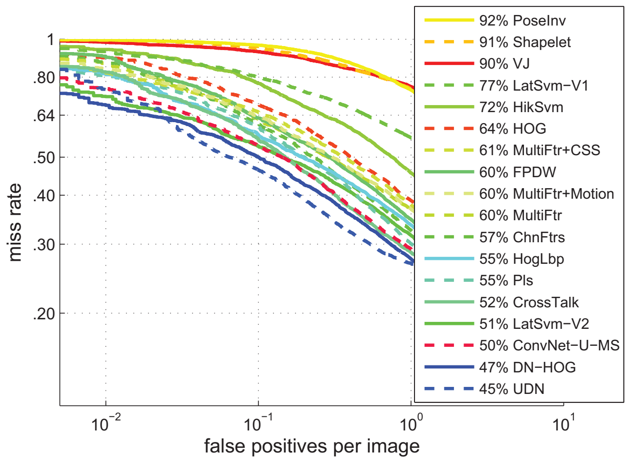


Figure 8. Experimental results on the ETH dataset.

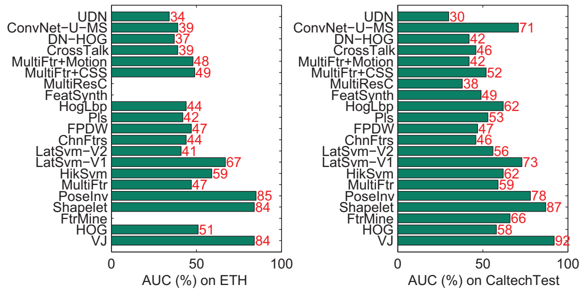


Figure 9. Comparisons of area under curve curve (AUC) on ETH and Caltech-Test. The results of MultiResC, Feat-Synth, and Ftr-Mine on ETH are not available.

4.2. 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 [34, 17, 44] on ETH); that is, using the INRIA training dataset in [5] 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. 8 shows the experimental results on ETH. Our UDN has the best performance on this dataset. Many studies (e.g., [24, 41]) 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 [44] is another measurement commonly used for evaluate the performance of pedestrian detection. Fig. 9 shows the average miss rate computed from AUC, which indicates that UDN also outperforms other sate-of-the-art methods under AUC. The results of MultiResC, FeatSynth, and FtrMine on the ETH dataset are not available.

5. 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 the best performance on publicly available datasets, outperforming the existing best performing approaches by 9% on the largest Caltech dataset. 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. This framework also has the potential for general object detection.

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) and National Natural Science Foundation of China (Project No. 61005057).

References

[1] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. In *ECCV*, 2010. 2, 5

[2] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using hough transforms. In *CVPR*, 2010. 1, 2

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

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

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2, 5, 7

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

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

[8] Y. Ding and J. Xiao. Contextual boost for pedestrian detection. In *CVPR*, 2012. 1, 6

[9] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*, 2012. 1, 2, 5

[10] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, 2010. 5

[11] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009. 1, 2, 5

[12] 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, 5

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

[14] D. Erhan, Y. Bengio, A.Courville, and P. Vincent. Visualizing higher-layer features of deep networks. Technical report, University of Montreal, 2009. 3, 4

[15] A. Ess, B. Leibe, and L. V. Gool. Depth and appearance for mobile scene analysis. In *ICCV*, 2007. 2, 5

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

[17] 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, 6, 7

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

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

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

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

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

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

[24] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 3, 5, 7

[25] 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, 3

[26] 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

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

[28] Z. Lin and L. Davis. A pose-invariant descriptor for human detection and segmentation. In *ECCV*, 2008. 5

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

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

[31] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008. 1, 2, 5

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

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

[34] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *CVPR*, 2012. 1, 2, 5, 6, 7

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

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

[37] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *ECCV*, 2010. 2, 5, 6

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

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

[40] 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

[41] 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. 7

[42] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *CVPR*, 2007. 5

[43] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *ICCV*, 2009. 2, 5

[44] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun. Pedestrian detection with unsupervised and multi-stage feature learning. In *CVPR*, 2013. 2, 5, 6, 7

[45] V. D. Shet, J. Neumann, V. Ramesh, and L. S. Davis. Bilattice-based logical reasoning for human detection. In *CVPR*, 2007. 2

[46] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for computing face similarities. In *ICCV*, 2013. 2

[47] 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

[48] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. 2

[49] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2):153–161, 2005. 1, 2, 5

[50] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR*, 2010. 2, 5

[51] X. Wang, X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *CVPR*, 2009. 1, 2, 5

[52] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *DAGM*, 2008. 5

[53] 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

[54] 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

[55] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 2

[56] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 2

[57] X. Zeng, W. Ouyang, and X. Wang. Multi-stage contextual deep learning for pedestrian detection. In *ICCV*, 2013. 2

[58] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010. 1, 2