Deep Learning Strong Parts for Pedestrian Detection

Yonglong Tian[1,3] Ping Luo[3,1] Xiaogang Wang[2,3] Xiaoou Tang[1,3]
[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]Department of Electronic Engineering, The Chinese University of Hong Kong
[3]Shenzhen Key Lab of Comp. Vis. & Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China
{ty014,pluo,xtang}@ie.cuhk.edu.hk, xgwang@ee.cuhk.edu.hk

Abstract

*Recent advances in pedestrian detection are attained by transferring the learned features of Convolutional Neural Network (ConvNet) to pedestrians. This ConvNet is typically pre-trained with massive general object categories (*e.g.* ImageNet). Although these features are able to handle variations such as poses, viewpoints, and lightings, they may fail when pedestrian images with complex occlusions are present. Occlusion handling is one of the most important problem in pedestrian detection. Unlike previous deep models that directly learned a single detector for pedestrian detection, we propose DeepParts, which consists of extensive part detectors. DeepParts has several appealing properties. First, DeepParts can be trained on weakly labeled data,* i.e. *only pedestrian bounding boxes without part annotations are provided. Second, DeepParts is able to handle low IoU positive proposals that shift away from ground truth. Third, each part detector in DeepParts is a strong detector that can detect pedestrian by observing only a part of a proposal. Extensive experiments in Caltech dataset demonstrate the effectiveness of DeepParts, which yields a new state-of-the-art miss rate of* 11.89%*, outperforming the second best method by* 10%*.*

1. Introduction

Pedestrian detection has been studied extensively in recent years [4, 7, 6, 1, 2, 29, 42] and has many applications such as video surveillance and robotics. While pedestrian detection has achieved steady improvements over the last decade, complex occlusion is still one of the obstacles. Referring to a recent survey [8], around 70% of the pedestrians captured in street scenes are occluded in at least one video frame. For example, the current best-performing detector SpatialPooling+ [27] attained 75% reduction of the average miss rate over the VJ detector [34] on Caltech [8] test set without occlusion. When heavy occlusions are present, it only attained 21% improvement over VJ [1].

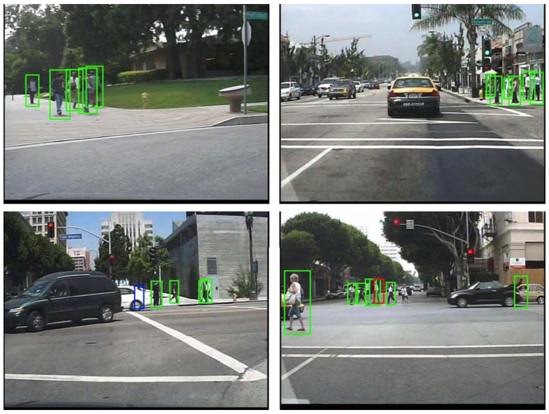


Figure 1. Pedestrian detection results on $Reasonable \cup HeavyOcclusion$ subsets, where pedestrians are larger than 49 pixels in height and have at least 20% body part visible. Green, red, and blue represent true positives, false positives, and missing positives, respectively.

Current pedestrian detectors for occlusion handling can be generally grouped into two categories, 1) training specific detectors for different occlusion types [36, 18] and 2) modeling part visibility as latent variables [10, 26, 22, 23, 9]. In the first category, constructing specific detector requires the prior knowledge of the occlusion types. For example, according to the statistics of the occlusion patterns in traffic scenes, [18] trained a series of biased classifiers for bottom-up and right-left occlusions. In the second category, [23, 22] divided pedestrian into several parts and inferred their visibility with latent variables. Although these methods achieved promising results, manually selecting parts may not be the optimal solution and may fail when handling pedestrian detection in other scenarios beyond traffic scenes, such as crowded scenes and market surveillance, where occlusion types change.

Inspired by [18], we introduce the idea of constructing a part pool that covers all the scales of different body parts

[1] http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/rocs/UsaTestRocs.pdf

1550-5499/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCV.2015.221

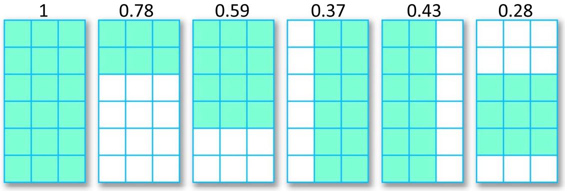


Figure 2. Complementary parts on Caltech pedestrian dataset and their normalized weights (*i.e.* importance).

and automatically choose important parts for occlusion handling. At the training stage, each part detector is learned by fine-tuning ConvNet features, which are pre-trained on ImageNet. At the testing stage, we design a shifting handling method within a ConvNet. This method handles the problem that positive proposal windows usually shift away from their corresponding ground truth bounding boxes. Moreover, the part selection is determined by data and the effectiveness of the part pool can be fully explored. Fig. 2 shows 6 body parts which are significant in the Caltech pedestrian dataset. They function complementarily to handle complex occlusions.

DeepParts has four main **contributions**. (1) We construct an extensive part pool where different complementary parts can be automatically selected in a data driven manner. The selected parts can be adopted to different scenarios or different datasets. (2) To our knowledge, we are the first to extensively explore how single part detector and their ensemble based on ConvNets contribute to pedestrian detection. In experiments, single part detector can achieve state-of-the-art performance while only observing a part of the proposal window, showing the robustness of DeepParts against occlusions. (3) We propose a novel method to handling proposal shifting problem. (4) We show that with complementary part selection, a new state-of-the-art miss rate of 11.89% can be achieved on the Caltech reasonable set.

1.1. Related Work

We review related works in three aspects.

Part-Based Pedestrian Detectors One stream of part-based approaches [20, 19, 9, 37] firstly trained part detectors in a fully supervised manner and then combined their outputs to fit a geometric model. For example, [20, 19] required part labels and were restricted to a limited number of manually-designed parts. Enzweiler *et al*. [9] utilized the depth and motion information to determine the occlusion boundaries. Wu *et al*. [37] assumed that the head of a pedestrian is visible and required a complex Bayesian framework to combine different components. In contrast, our method does not need part annotations and can automatically select complementary parts (components of human body) from a large part pool.Another stream of part-based models focused on unsupervised part mining, which does not require part labels. Felzenszwalb *et al*. [10] proposed Deformable Part Model (DPM), which learned a mixture of local templates for each body part to handle pose variations. Lin *et al*. [16] proposed a promising and effective framework by incorporating DPM into And-Or graph. Recently, Girshick *et al*. [13] reformulated DPM as ConvNet. DPM needs to handle complex configurations while our method is much simpler.

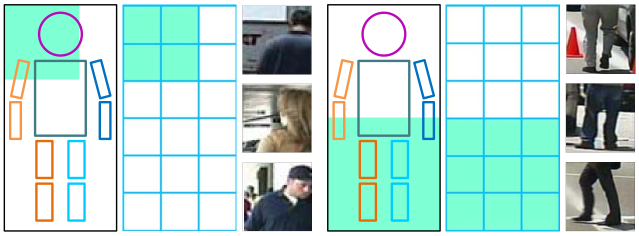
Occlusion Handling Some recent works [18, 31, 24] focused on handling specific types of pedestrian occlusion. For example, the Franken-classifiers [18] learned a small set of classifiers, where each one accounts for a specific type of occlusion. In this work, we extend this idea by constructing an extensive part pool. Unlike [18] that the parts were pre-defined, our complementary parts are automatically determined by data and may vary in different scenarios or datasets. In [31, 24], occlusions caused by overlaps between two pedestrians were handled. Specifically, Tang *et al*. [31] proposed a pedestrian detector tailored to various occlusion levels, while Ouyang *et al*. [24] employed a probabilistic framework to model the relationship between the configurations estimated by single- and multi-pedestrian detectors. With the large part pool, our method can cover more occlusion patterns.

Deep Models Deep learning methods can learn high level features to aid pedestrian detection. For instance, Ouyang *et al*. [23, 22] introduced a part deformation layer into deep models to infer part visibility. By introducing switchable layers to learn both low-level features and high-level semantic parts, SDN [17] achieved further improvement. Because the receptive field of higher layers in ConvNet is large (sometimes covers most of the input patch), modeling part visibility in a single ConvNet as these methods can not explicitly learn visual patterns for each part and may suffer from part co-adaption. Tian *et al*. [32] modeled detection task together with attribute prediction tasks within a single deep model. Finally, Hosang *et al*. [14] demonstrated the effectiveness of the R-CNN pipeline [12] in pedestrian detection and achieved top performance on Caltech [8] and KITTI[11]. We follow this framework to train our strong part detectors. Moreover, unlike Part-Based R-CNN [41], our DeepParts does not need part annotations in training.

Another series of methods [7, 6, 21, 42, 43] focusing on Channel Features and feature selection also achieved state-of-the-art performance for pedestrian detection, but they are not specially designed for occlusion handling.

2. Training Part Detectors

We take several steps to build our part-based pedestrian detector. Firstly, we construct a part pool, where the parts cover the full body of pedestrian at different positions and scales. We then learn a detector for each of the part. A



(a) part (1,1,2,2,1)

(b) part (1,5,3,2,35)

Figure 3. Part prototype examples, (x, y, w, h, i) is defined in Eqn.(2) (a) head-left-shoulder part with 2 grids in height and width; (b) leg part with 2 grids in height and 3 grids in width.

method is further designed to handle shifting problem of proposal windows. Finally, we infer the full body score over complementary part detectors.

2.1. Part Pool

Occlusions may present at different body parts and have various patterns. For instance, the left- or right-half body may be occluded by a tree, and the lower-half body may be occluded by a car. Thus, we construct an extensive part pool, containing various semantic body parts.

We consider pedestrian as a rigid object and define a human body grid of $2m \times m$, where $2m$ and m indicate the numbers of cells in horizontal and vertical direction, respectively. Each cell is a square and has equal size. Furthermore, we ensure each part to be a rectangle. The scales for parts are defined as

$$S = \{(w, h) | W_{min} \le w \le m, H_{min} \le h \le 2m, \\ w, h \in \mathbb{N}^{+}\}, \quad (1)$$

where w and h indicate the width and height of a part respectively, in terms of the number of cells they contain. W_{min} and H_{min} are used to avoid subtle part since we focus on middle-level semantic part. Then, for each $(w, h) \in S$, we slide a $h \times w$ window over the human body grid with step size s, to generate parts at different positions. The entire part pool could be expressed as follows

$$P = \{(x, y, w, h, i) | x, y \in \mathbb{N}^{+}, (w, h) \in S, i \in I\}, \quad (2)$$

where x and y are the coordinates of the top-left cell in the part and i is a unique id. Specifically, the part representing the full body is defined as $(1, 1, m, 2m, i_{full})$.

Large m results in a large part pool, which may cause more computations in the training and testing stages. Also, small values of W_{min} and H_{min} result in subtle parts, such as $W_{min} = 0.1 \times m$. To avoid the above issues, we have $m = 3$, $W_{min} = 2$, $H_{min} = 2$, and $s = 1$ in our implementation, resulting in a part pool with 45 prototypes. Two examples regarding the parts of head-left-shoulder and leg are shown in Fig.3.

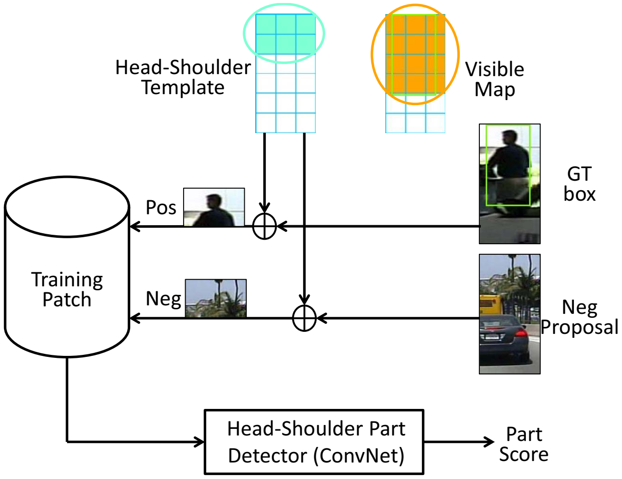


Figure 4. Using head-shoulder part prototype as an illustration for training part detectors. (1) data: we extract corresponding regions within negative proposals as negative training samples; we compute the visible map of each ground truth box, and extract the corresponding region as a positive sample only if the part template is fully covered by the visible map. (2) ConvNet: a part detector can be any deep models, such as AlexNet, Clarifai and GoogLeNet.

2.2. Training

For each part, we train an independent ConvNet classifier. This leads to totally $|P|$ models, where $| \cdot |$ is the counting operator.

Training Data The size of training dataset is crucial for ConvNets. In our experiments, we use Caltech dataset [8], which is the largest pedestrian benchmark that consists of ∼250k labeled frames and ∼350k annotated bounding boxes. Instead of following the typical *Reasonable* setting, which uses every 30^{th} image in the video and has ∼1.7k pedestrians for training, we utilize every frame and employ ∼50k pedestrian bounding boxes as positive training patches. Proposals are obtained by LDCF [21]. Negative patches are the proposed windows that have IoU < 0.5 with the ground truth bounding boxes.

Part Specific Patch Generation As shown in Fig.4, we take the head-shoulder detector as an example, to illustrate how to generate the training data. (1) Given the definition of a part, we consider the corresponding region within a negative proposal (*i.e.* patches containing objects/backgrounds other than pedestrians) as the negative sample. (2) Each pedestrian is annotated with two bounding boxes (BBs), which denote the visible part B_{vis} (in green) and full body B_{full}, respectively. We divide the full body B_{full} into $2m \times m$ cells and compute the visible ratio of each cell. Then we obtain the visible map by thresholding on the visible ratio. If the visible cells of a ground truth can cover the cells of a part, we extract the corresponding region of this part as a positive sample. In our implementation, the threshold for computing the visible map is 0.4.

Pre-training It has been demonstrated in R-CNN [12] that fine-tuning a pre-trained ConvNet on ImageNet classification task on object detection and segmentation data can significantly improve the performance. Particularly, the parameters learned at the pre-training phase are directly used as initial values for the fine-tuning stage. Similar strategy can be directly adopted to fine-tune the generic ConvNet image classification models for part recognition. The main disparity between the pre-training and fine-tuning tasks is the type of input data. Image classification task employs the full image or the entire object as input, which contains rich context information while part recognition task can only observe a middle-level patch of a part.

To understand how to narrow the above disparity, we investigate three popular deep models and three pre-training strategies as below. Three deep models are **AlexNet** [15], **Clarifai** [39], and **GoogLeNet** [30], which are the best-performing models of the ImageNet [5] classification challenge in the past several years. AlexNet and Clarifai have ∼60 million parameters and share similar structures, while GoogLeNet uses $12\times$ fewer parameters but is much deeper than the first two models. Our framework is flexible to incorporate other generic deep models.

Three pre-training strategies include (1) no pre-training, *i.e.* randomly initializing model parameters with Gaussian distribution (**strategy 1**), (2) pre-training the deep models by using the ImageNet training data with image-level annotations of 1000 classes, *i.e.* taking the full images as input, such as [14] (**strategy 2**), and (3) pre-training the deep models by using ImageNet training data with object-level annotations of 1000 classes [25], *i.e.* taking the cropped object patches as input (**strategy 3**).

Fine-tuning For each part prototype, we use the part specific patches to fine-tune our part detectors. We replace the last $d \times n$ classification layer with $d \times 2$ randomly initialized part classifier (d and n indicate the feature dimension and the number of pre-training classes, respectively). In our implementation, we uniformly sample 16 positive and 48 negative windows to construct a mini-batch. Experiments show that fine-tuning for 10000 iterations with a learning rate of 0.001 is sufficient to converge.

2.3. Handle Shifting in Deep Model

In pedestrian detection, the localization qualities (*i.e.* whether the window is tight or not) of the proposals are important for the recognition stage. Pedestrian detectors usually suffer from poor localization quality of the proposals. As reported in [14], the best proposal method *SpatialPooling+* [27] recalls 93% pedestrians with 0.5 IoU threshold while only recalls 10% with 0.9 IoU threshold. Shifting is one of the major reasons that cause low IoU. As shown in Fig 5(a), shifting a ground truth bounding box by 10% on horizontal or vertical direction leads to 0.9 IoU, which is still a proposal of high quality. However, shifting on both directions leads to 0.68 IoU, such that critical parts are missing, hindering the stages of feature extraction and classification. Except the full body shifting, each body part may also shift and different parts of the same pedestrian may shift towards different directions. In our framework, the positive training samples for each part detector are well aligned while the testing proposals may shift at all directions. Thus, handling shifting for both the full body and each part is necessary.

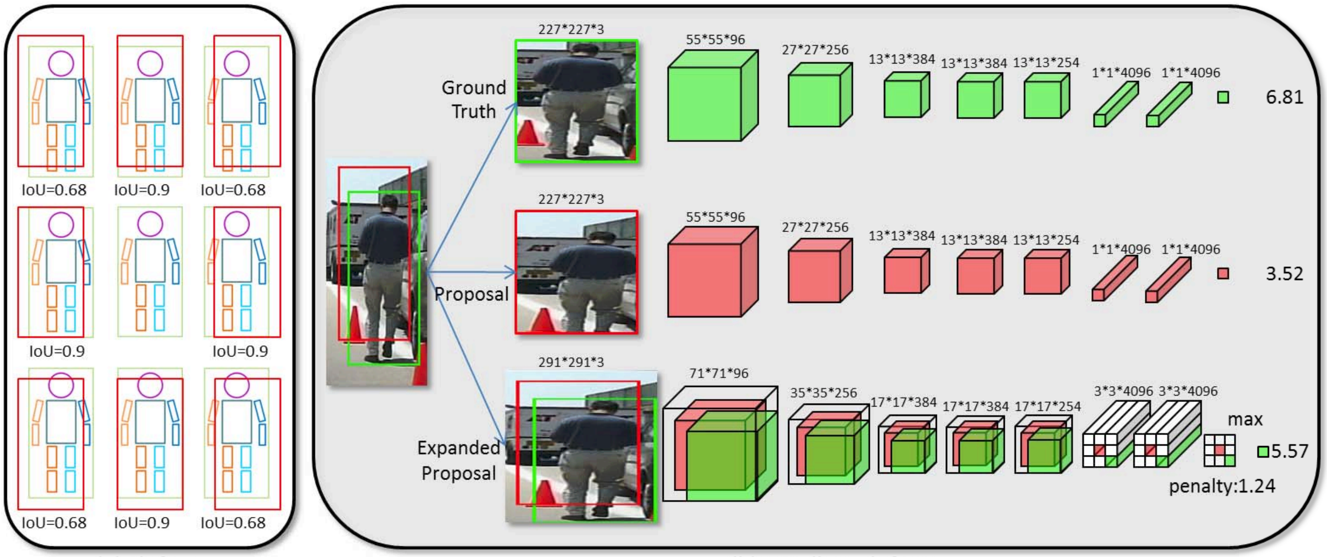
type	**filter/stride**	**output map size**	
		no extension 227×227	extended by $32n$ $(227+32n) \times (227+32n)$
conv1	$11 \times 11/4$	55×55	$(55+8n) \times (55+8n)$
pool1	$3 \times 3/2$	27×27	$(27+4n) \times (27+4n)$
conv2	$5 \times 5/1$	27×27	$(27+4n) \times (27+4n)$
pool2	$3 \times 3/2$	13×13	$(13+2n) \times (13+2n)$
conv3	$3 \times 3/1$	13×13	$(13+2n) \times (13+2n)$
conv4	$3 \times 3/1$	13×13	$(13+2n) \times (13+2n)$
conv5	$3 \times 3/1$	13×13	$(13+2n) \times (13+2n)$
pool5	$3 \times 3/2$	6×6	$(6+n) \times (6+n)$
conv6	$6 \times 6/1$	1×1	$(1+n) \times (1+n)$
conv7	$1 \times 1/1$	1×1	$(1+n) \times (1+n)$
conv8	$1 \times 1/1$	1×1	$(1+n) \times (1+n)$

Table 1. Fully convolutional structure of AlexNet. We turn the original fully connected layer $fc6(4096)$, $fc7(4096)$ and $fc8(2)$ into $conv6(1\times1\times4096)$, $conv7(1\times1\times4096)$,$conv8(1\times1\times2)$.

A straight forward way to handle this problem is that we crop multiple patches around each proposal with jitter, then feed the cropped patches into the deep model and choose the highest or averaged score with penalty as the detection score. However, this method would increase the testing time by k times, where k is the number of cropped patches for each proposal.

To reduce the testing computation, we firstly reformulate the generic ConvNet models with fully connected layer as fully convolutional neural networks, which does not require fixed input size and can process multiple neighboring patches via only one forward pass. In this case, the input size of the fully convolutional ConvNet can be changed. We take the AlexNet as an example, the original input size of which is 227×227. As illustrated in Table 1, after reformulating $fc6$, $fc7$, $fc8$ as $conv6(1 \times 1 \times 4096)$, $conv7(1 \times 1 \times 4096)$, $conv8(1 \times 1 \times 2)$, the fully convolutional AlexNet is able to receive an expanded input size because the convolution and pooling operations are unrelated to the input size. Since the step size of receptive field for the classification layer is 32, the expanded input should be $(227+32n) \times (227+32n)$ in order to keep the forward procedure applicable, where n indicates expanded step size and is a non-negative integer.

Given a proposed part patch (x_{min}, y_{min}, w, h) and n, the expanded cropping patch is $(x'_{min}, y'_{min}, w', h')$, where



(a) Shifting

(b) Handling Shifting

Figure 5. **Best viewed in color.** (a) shows how rapidly IoU will decrease with little shifting on horizontal and vertical orientation. (b) shows how to handle shifting problem in AlexNet. A true positive proposal which shifts 14.1%, namely 32/227, on both horizontal and vertical side is scored as 3.52 while the corresponding ground truth is scored as 6.81. With the neighboring search and penalization, our detector adjusts the score value to 5.40.

$$x'_{min} = x_{min} - \frac{16n}{227} \times w,\ \ y'_{min} = y_{min} - \frac{16n}{227} \times h,$$
$$w' = (1 + \frac{32n}{227}) \times w,\ \ h' = (1 + \frac{32n}{227}) \times h. \tag{3}$$

Then we resize the patch to $(227 + 32n) \times (227 + 32n)$ and feed it into the fully convolutional AlexNet. As a result, $(1+n) \times (1+n)$ neighboring 227×227 patches are evaluated simultaneously while the expanded scale keeps the same as the proposal scale. The final output of $conv8$ can be viewed as a $(1+n) \times (1+n)$ score map S and each score corresponds to a 227×227 region. The final score of the part patch is defined as

$$s = \max_{1 \le i,j \le n+1} \{S_{i,j} - P_{i,j}\} \tag{4}$$

where $P_{i,j}$ is a penalty term with respect to relative shifting distance from the proposed part box and is defined as

$$\begin{aligned} P_{i,j} = a \times (|i - \frac{n+2}{2}| + |j - \frac{n+2}{2}|) \times \frac{32}{227} \\ + b \times (|i - \frac{n+2}{2}|^2 + |j - \frac{n+2}{2}|^2) \times (\frac{32}{227})^2 \end{aligned} \tag{5}$$

where a is the single orientation shifting penalty weight (we give the same weight on both horizontal and vertical orientations), and b is a geometrical distance penalty weight.

In our implementation, we have $n = 2$ for all parts and search the values of a, b for each part by a 6-fold cross validation on training set. Fig.5 (b) shows an example of the full body part detector with 9 neighboring patches evaluated, where $a = 2$ and $b = 10$. Shifting handling is a kind of context modeling which keeps scale invariant, while simply cropping larger region with padding and resizing to 227×227 bring a scale gap between the training and testing stages.

3. Parts Complementarity

For each part, we directly use the output of its ConvNet detector as the visible score instead of stacking a linear SVM on the top as the R-CNN framework [12]. We find that appending a SVM detector for mining hard negatives does not show significant improvement over directly using the ConvNet output, especially for GoogLeNet. This may due to the fact that the training proposals generated by LDCF[21] are already hard negatives. Thus, we safely remove the SVM training stage to save computation time.

Then we employ a linear SVM to learn complementarity over the 45 part detector scores. To alleviate the testing computation cost, we simply select 6 parts with highest value of the SVM weight, yielding approximate performance. Experiments show that the performance improvement mainly benefits from the part complementarity.

4. Experiments

DeepParts is evaluated on the Caltech dataset [8], using subsets set00-set05 for training and set06-set10 for testing. We strictly follow the evaluation protocol of [8], measuring the log average miss rate over nine points ranging from 10^{-2} to 10^0 False-Positive-Per-Image. Three subsets are considered for testing evaluation (*Reasonable*, *Partial occlusion* and *Heavy occlusion*).

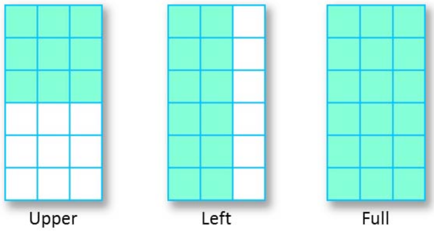


Figure 6. Part prototypes of upper, left and full body.

Fine-tuning data	upper	left	full
Every 30th image	41.19	46.96	33.01
Every 10th image	38.25	46.36	30.45
Every 5th image	36.63	41.59	23.83
Every 3rd image	34.60	39.69	23.18
Every image	34.11	37.83	21.19

Table 2. Log-average miss rate (%) on Caltech-Test of upper, left and full body parts with respect to data volume. All results are obtained by fine-tuning from AlexNet which adopted image-level pre-training strategy (*i.e.* **strat.2**).

• *Reasonable* subset. Pedestrians are larger than 49 pixels in height and have at least 65 percent visible body parts. Reasonable subset is considered as a more representative evaluation than overall performance on all pedestrians and is the most frequent evaluation setting. Without special illustration, we use *Reasonable* as our default setting to compare performance.

• *Partial* and *heavy occlusion* subsets where pedestrians are larger than 49 pixels in height and have $1-35$ and $36-80$ percent occluded body parts, respectively.

Because of page limit, we choose three representative parts for illustration. The selected parts are upper, left, and full body respectively, and are shown in Fig.6. For each part, we directly use the part detector output (without SVM training) as the whole patch score to evaluate the performance of single part detector. The complete results are included in the **supplementary material**.

4.1. Evaluation of data volume

To investigate how data volume influences the fine-tuning performance of part detectors, we compare the performance obtained by fine-tuning AlexNet with five different sets of data. Here, AlexNet is pre-trained on ImageNet image-level data (*i.e.* **strat.2**). We present results in Table 2. The average miss rate of each part detector shows a decreasing pattern when incorporating more image frames. For example, the upper, left, and full body detector achieve 7.08%, 9.13% and 11.82% improvements, respectively, when the data volume is increased by 30×. The three models are still unsaturated though all training frames have been utilized.

Body Part	no SH	SH	SVM
Upper	26.02	23.93	25.11
Left	29.21	27.43	27.74
Full	16.43	15.41	16.17

Table 3. Effectiveness of shifting handling via a comparison with no shifting handling and appending SVM on the top. We achieve the best value for SVM by greedily search the value of C and overlap thresholds for positive and negative samples.

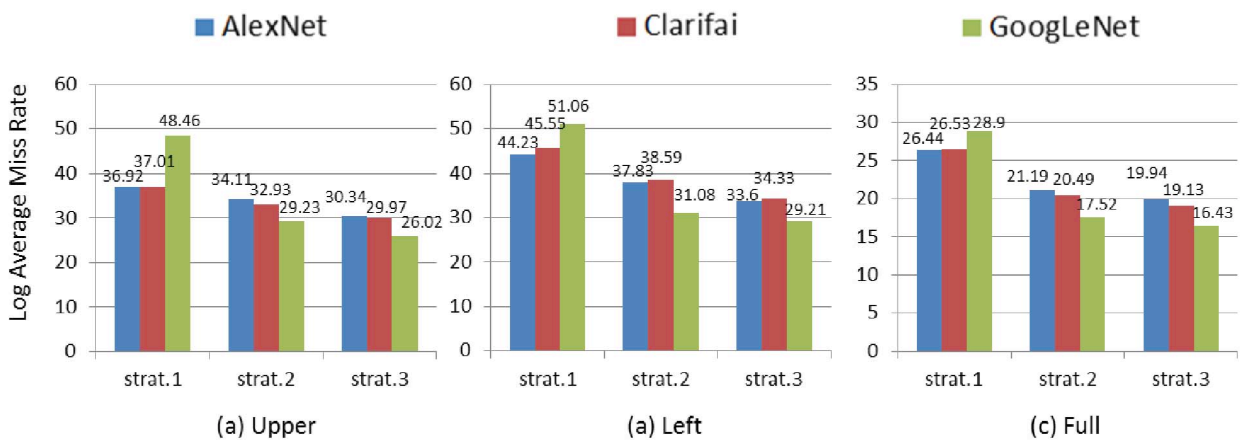
4.2. Evaluation of models and pre-training

We evaluate the single detector performance of upper, left, and full body parts with different contemporary deep architectures. All the deep models are pre-trained with three strategies, including (1) random initialization (**strat.1**), (2) image-level pre-training (**strat.2**), and (3) object-level pre-training (**strat.3**). As shown in Fig.7, GoogLeNet outperforms AlexNet and Clarifai over all the three body parts with ImageNet pre-training. Fine-tuning an object-level pre-trained GoogLeNet solely on the upper body part can yields 26.02% miss rate, which is already close to the strong LDCF proposals. When the full body is utilized, the miss rate surprisingly reduces to 16.43%, which is the best result that ever reported on Caltech *reasonable* subset. Besides, it is noticeable that GoogLeNet is inferior to AlexNet and Clarifai with random initialization. We believe it is because of the model structure, where GoogLeNet are much deeper and needs more iterations to converge when training from scratch.

For all three models, random initialization (**strat.1**) of network parameters leads to the worst performance. For full body part, object-level pre-training (**strat.3**) strategy shows around 1.2% superiority over image-level pre-training (**strat.2**) strategy. However, as for upper part and left part, this gap increases to around 4% for AlexNet and Clarifai, and 2.5% for GoogLeNet. This may reveal the fact that pre-training on object-level data are more capable to model mid-level part variations than on image-level data.

4.3. Evaluation of handling shifting

To understand the effectiveness of shifting handling (**SH**), we compare the results of shifting handling with that of directly utilizing the single patch score given by the last classification layer (**no SH**) and that of appending a SVM on top to mine hard negatives (**SVM**). The results are collected in Table 3, which shows that Shifting Handling consistently achieves the lowest miss rate over upper, left, and full body parts. Besides, SVM improves 1.47% over network output for the left part but only improves 0.26% for full body part, while the improvement of shifting handling shows a much slower fading pattern as with lower miss rate. Experiments also reveals that handling shifting pro-



(a) Upper (a) Left (c) Full

Figure 7. Log-average miss rate (%) on Caltech-Test reasonable subset. The AlexNet, Clarifai and GoogLeNet are represented by blue, red and green, respectively. Strategy 1, 2, 3 indicate random initialization, image-level pre-training and object-level pre-training, respectively.(a) upper body part. (b) left body part. (c) full body part.

vides more benefits for smaller parts, *i.e.* the average improvement for 2×2 parts is 4.3% but it drops to 2.1% for 3×3 parts. This is consistent with the fact that smaller parts are more flexible to shift.

4.4. Overall evaluation

We report the final results on Caltech-Test in two aspects.

In the first aspect, we investigate the overall pipeline by adding each component step-by-step, which is summarized in Table 4. The strong LDCF [21] has 24.80% miss rate. Single full body part detector improves 3.61% by fine-tuning AlexNet, which is pre-trained on image-level data. Pre-training GoogLeNet rather than Alex-Net improves miss rate by 3.67%. Changing the pre-training strategy from image-level annotation to bounding box level annotation also improves 1.09%. Combining all parts detector (*i.e.* 45 in our implementation. All of them are fine-tuned on GoogLeNet which employed bounding box data for pre-training.) further reduces the miss rate by 3.31%. By adding shifting handling for each part detector, the final average miss rate on the *reasonable* subset is 11.89%.

In order to reduce testing time, we picked 6 part detectors from the entire part pool, decided by the top 6 weights of the ensemble SVM. As illustrated in Table 4, the ensemble of the 6 part detectors reaches 12.31%, which shows that the selected models offer the major improvement of the whole part pool. To understand whether the improvement comes from model ensemble itself or learning part complementarity, we construct another two experiments, 1) by combining 6 best independent parts and 2) by combining 6 full body detectors which are fine-tuned on AlexNet, Clarifai, and GoogLeNet with image and bounding box pre-training strategies, respectively. The weights for combination are learned by SVM. As shown in Table 4, simply combining the 6 best-performing part detectors reduces the performance, *i.e.* the miss rate increases 2.97%. Besides, ensemble of AlexNet, Clarifai, and GoogLeNet only achieves 15.5% miss rate. This reveals that the part complementarity is the major reason for ensemble improvement. The 6 selected parts are given in Fig.2.

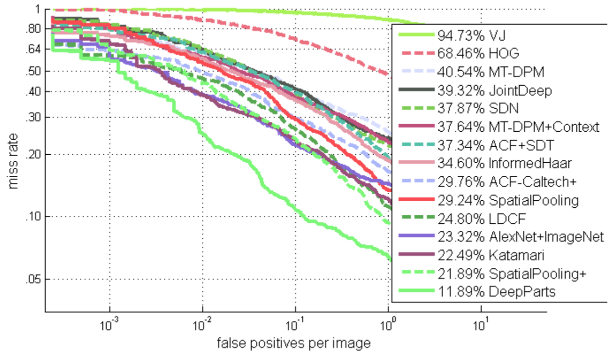


Figure 8. Average miss rate on *reasonable* subset.

In the second aspect, we compare the overall result of DeepParts with existing best-performing methods , including VJ [33], HOG [4], MT-DPM [38], MT-DPM+Context [38], JointDeep [23], SDN [17], ACF+SDT [28], InformedHaar [42], ACF-Caltech+ [21], SpatialPooling [27], LDCF [21], AlexNet+ImageNet [14], Katamari [3], SpatialPooling+ [27]. Besides, we also compare the DeepParts with all existing deep models on reasonable subset, including ConvNet [29], DBN-Isol [22], DBN-Mut [26] and MultiSDP [40].

Fig.8, Fig.9, and Fig.10 report the results on *reasonable*, *partial occlusion*, and *heavy occlusion* subsets, respectively. DeepParts outperforms the second best method (SpatialPooling+ [27]) by 10 percent on the reasonable subset, which is a large margin. Besides, DeepParts improves the average miss rate on partial and heavy occlusion subsets by 19.32% and 14.23%, showing its potential to handle occlusion at different levels.

Deep Models Fig.11 shows that DeepParts achieves the

detection pipeline	LDCF	AlexNet	AlexNet	image	parts	shifting	6 parts	top 6	A,C,G
			to GoogLeNet	to box	ensemble	handling		parts	ensemble
miss rate (%)	24.80	21.19	17.52	16.43	13.12	11.89	12.31	15.28	15.50
improvement (%)		+3.61	+3.67	+1.09	+3.31	+1.23			

Table 4. Ablation study of our pipeline.

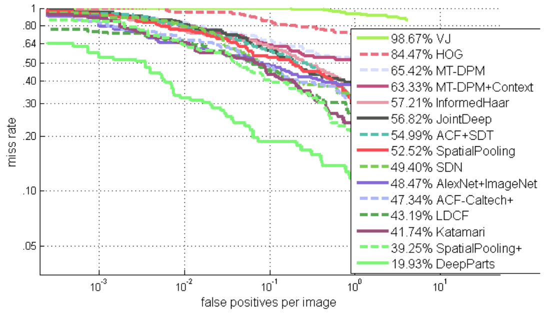


Figure 9. Average miss rate on *Partial Occlusion* subset.

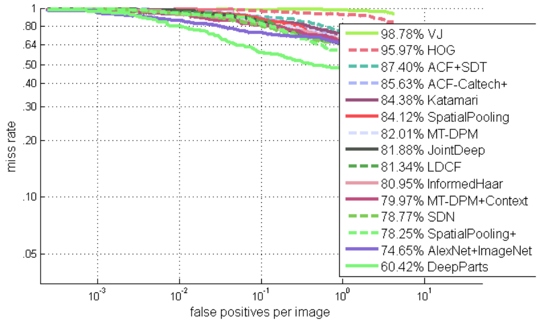


Figure 10. Average miss rate on *Heavy Occlusion* subset.

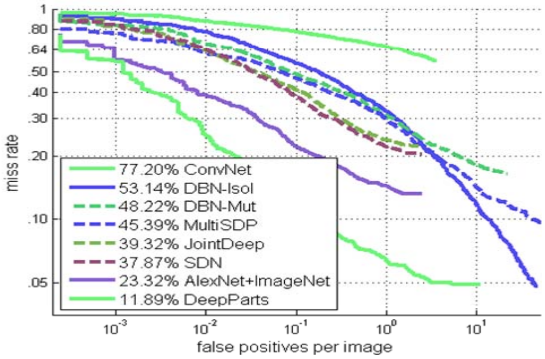


Figure 11. Comparison between deep models.

lowest miss rate among all deep models. For example, it outperforms DBN-Isol, DBN-Mut and JointDeep, which are also based on part modeling, by 41, 36 and 26 percent. In addition, DeepParts reduces the miss rate by 11 percent over AlexNet-ImageNet, which is also fine-tuned on the ImageNet pre-trained model. However, AlexNet-ImageNet fails to model the part visibility and thus is hard to handle occlusion.

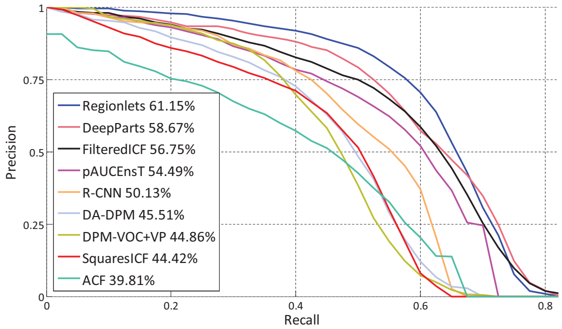


Figure 12. Results on KITTI with moderate subset.

4.5. KITTI

To test the generalization ability and verify whether the 6 complementary parts can be transferred to other street pedestrian detection dataset, we test our DeepParts on KITTI [11]. We do not use the training data of KITTI and all components are trained on Caltech. DeepParts achieves promising results, *i.e*., 70.49%, 58.67% and 52.78% AP on easy, moderate, and hard subsets respectively. Fig.[11] represents the results on moderate subset (pedestrians are no less than 25 pixels in height). The best detector Regionlets [35] classified both the cyclists and pedestrians. The additional supervision improved its performance of pedestrian detection. It outperforms DeepParts by 2.48%. Without cyclists as supervision, DeepParts is the best-performing detector based on ConvNet, and surpasses R-CNN by 8.54%.

5. Conclusion

In this paper, we proposed DeepParts to improve the performance of pedestrian detection by handling occlusion with an extensive part pool, showing significant superiority over previous best-performing models. The DeepParts can also be treated as a cascade stack over other pedestrian detectors to further improve performance. Future work lies towards model compression, such as incorporating all part detectors into one ConvNet.

Acknowledgement Ping Luo* is supported by the National Natural Science Foundation of China (61503366). This work is also partially supported by the National Natural Science Foundation of China (91320101, 61472410).

*indicates the corresponding author.

References

[1] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *CVPR*, 2012.

[2] R. Benenson, M. Mathias, T. Tuytelaars, and L. V. Gool. Seeking the strongest rigid detector. In *CVPR*, 2013.

[3] R. Benenson, M. Omran, J. Hosang, , and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV, CVRSUAD workshop*, 2014.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*. 2005.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *TPAMI*, 2014.

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

[8] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *TPAMI*, 2012.

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

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

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

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

[14] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] L. Lin, X. Wang, W. Yang, and J. Lai. Discriminatively trained and-or graph models for object shape detection. In *TPAMI*, 2015.

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

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

[19] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV*, 2004.

[20] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *TPAMI*, 2001.

[21] W. Nam, P. Dollár, and J. H. Han. Local decorrelation for improved pedestrian detection. In *NIPS*, 2012.

[22] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *CVPR*, 2012.

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

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

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

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

[27] S. Paisitkriangkrai, C. Shen, and A. v. d. Hengel. Pedestrian detection with spatially pooled features and structured ensemble learning. *arXiv*, 2014.

[28] D. Park, C. Zitnick, D. Ramanan, and P. Dollar. Exploring weak stabilization for motion feature extraction. In *CVPR*, 2013.

[29] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*. 2013.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv*, 2014.

[31] S. Tang, M. Andriluka, and B. Schiele. Detection and tracking of occluded people. *IJCV*, 2014.

[32] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *CVPR*, 2015.

[33] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004.

[34] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 2005.

[35] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, 2013.

[36] C. Wojek, S. Walk, S. Roth, and B. Schiele. Monocular 3d scene understanding with explicit occlusion reasoning. In *CVPR*, 2011.

[37] 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.

[38] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Li. Robust multi-resolution pedestrian detection in traffic scenes. In *CVPR*, 2013.

[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*. 2014.

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

[41] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014.

[42] S. Zhang, C. Bauckhage, and A. Cremers. Informed haar-like features improve pedestrian detection. In *CVPR*, 2014.

[43] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *CVPR*, 2015.