Multi-Stage Contextual Deep Learning for Pedestrian Detection

Xingyu Zeng Wanli Ouyang Xiaogang Wang
Department of Electronic Engineering, The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
{xyzeng, wlouyang, xgwang}@ee.cuhk.edu.hk

Abstract

Cascaded classifiers[1] *have been widely used in pedestrian detection and achieved great success. These classifiers are trained sequentially without joint optimization. In this paper, we propose a new deep model that can jointly train multi-stage classifiers through several stages of back-propagation. It keeps the score map output by a classifier within a local region and uses it as contextual information to support the decision at the next stage. Through a specific design of the training strategy, this deep architecture is able to simulate the cascaded classifiers by mining hard samples to train the network stage-by-stage. Each classifier handles samples at a different difficulty level. Unsupervised pre-training and specifically designed stage-wise supervised training are used to regularize the optimization problem. Both theoretical analysis and experimental results show that the training strategy helps to avoid overfitting. Experimental results on three datasets (Caltech, ETH and TUD-Brussels) show that our approach outperforms the state-of-the-art approaches.*

1. Introduction

Pedestrian detection is one of the fundamental problems in computer vision with numerous important applications. Due to various challenges, such as variations of views, poses, lightings and occlusions, pedestrian detection is difficult and unlikely to be well solved with one simple holistic classifier. For example, the visual cue of a pedestrian with a side view is different from that with a frontal view. It is difficult for a single detector to well capture the two visual cues simultaneously. In order to handle the complex appearance variation of pedestrians, many approaches choose a group of classifiers to make the pedestrian versus non-pedestrian decision stage by stage [49, 2]. Different classifiers take care of different portions of samples. The cascaded classifiers are usually trained sequentially. Hard samples which cannot be well classified at early stages are used to train classifiers at later stages.

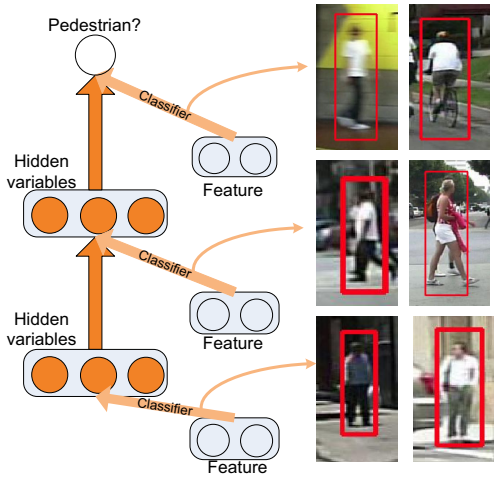

Figure 1. Our multi-stage contextual deep model. This architecture can deal with complex distributed samples using multiple stages of classifiers. In each stage, a classifier deals with samples at a different difficulty level.

Intuitively, it is desirable to jointly optimize these classifiers, since they create synergy through close interaction. Moreover, although early classifiers cannot make final decisions on hard samples, their output provides contextual information to support decisions at later stages. However, with too many parameters and relatively few training samples, the classifiers easily overfit training data. To jointly train a large number of classification parameters, we propose a deep model that can learn these classifiers together and keep the training process from overfitting in the meanwhile. Figure 1 shows its architecture, in which the outputs of classifiers are represented as hidden nodes in each layer. Each layer takes images features and the output of its previ-

[1]Cascading means concatenation of multiple classifiers and the output of a classifier is used as additional input to the next classifier in the cascade. Specifically in detection literature, cascaded classifiers often indicate early rejecting samples. Our proposed deep model cascades classifiers but does not early rejects samples. We choose the term multi-stage for our model to avoid confusion.

ous layer as input, then outputs new decision scores.

The contribution of this paper is four-folds:

- The group of classifiers in the deep model choose training samples stage by stage. The training is split into several back-propagation (BP) stages. Due to the design of our training procedure, the gradients of classifier parameters at the current stage are mainly influenced by the samples misclassified by the classifiers at the previous stages. At each BP stage, the whole deep model has been initialized with a good starting point learned at the previous stage and the additional classifiers focus on the misclassified hard samples.
- The group of classifiers are jointly optimized. At each BP stage, classifiers at the previous stages jointly work with the classifier at the current stage in dealing with misclassified samples.
- We propose a training procedure that helps to avoid overfitting. Both unsupervised pre-training and specifically designed stage-wise supervised training are used to regularize the optimization problem. It is different than standard BP, which optimizes the network as whole. With standard BP, an easy training sample can influence classifiers at any stage, since these stages are not distinguished or separately trained. The search parameter space is huge and it is easy to overfit.
- Existing cascaded classifiers only pass a single score to the next stage, while our deep model keeps the score map within a local region and it serves as contextual information to support the decision at the next stage.

2. Related work

Newly designed features constantly improve detection performance. Seminal works on feature design include Haar-like features [49], scale-invariant feature transform (SIFT) [33], dense SIFT [48], histogram of gradients (HOG) [6], gradient histogram [12], local binary pattern (LBP) [1], color histogram [51], and color self similarity (CSS) [51]. Other cues like depth [20], segmentation [16, 18] and motion [7] also improve the detection performance. We use the modified HOG and CSS features.

Most approaches consider pedestrian detection as a classification task by scanning an image with sliding windows whose sizes are changeable. A large number of generative and discriminative classification approaches have been developed. The generative methods, such as [17], calculate the probability of a window enclosing a pedestrian. The discriminative classifiers, such as boosting classifiers [49, 12, 55, 10, 11, 15] and SVM [24, 56, 36, 24, 56], seek for parameters to separate positive and negative samples.

Part-based models have shown to be effective in object detection and recognition [24, 23, 4]. The deformable part based model in [24, 56] is able to detect objects with some pose changes. This model is then extended to cascaded classifiers in [23] to boost the computational speed. Poselet in [4] is able to handle the appearance variation of parts.

As pedestrians have diverse appearance, mixture of parts has been used in many approaches [18, 8, 4]. Mixture models train classifiers through supervised or unsupervised clustering. Differently, cascaded classifiers are trained with misclassified training samples stage by stage.

Cascaded classifiers have two advantages: (1) they can provide piecewise linear classification hyperplanes, and (2) they help to save the computational load of sliding window object detection. Therefore, many approaches have used cascade [49, 12, 55, 10, 3, 9, 2, 23]. While the cascade structure has worked well in many fields, the hard thresholding for each cascaded classifier discards a lot of information collected at each stage classifier. To avoid such disadvantages, recent detection approaches have used soft cascade [3, 10, 2, 9, 12], which collects the classification scores extracted by each stage of classifiers and then combines the classification scores for the final decision. However, soft cascade still learns the classifiers stage by stage without joint optimization. Classifiers at different stages cannot cooperate with each other in the training procedure.

Recently, deep models have been successfully applied in hand written digit recognition [27, 26, 31, 37], object segmentation [34, 35], face recognition [5, 47, 57], scene understanding [43, 21], object detection [40] [41] [38] [39] and recognition [28, 30, 29, 46, 25]. Sermanet et al. [46] unsupervised learned multi-stage features with a deep model. However, they did not add an extra classifier at each stage and classification scores were not passed between stages as contextual information. Hinton et al. [26] proved that adding a new layer, if done correctly, creates a model that has a better variational lower bound on the log probability of the training data. However, the connection between deep models and multi-stage classifiers is unknown. This paper is complementary to the recent deep models in that we have built the connection between deep models and multi-stage classifiers, such that the cascaded classifiers can be jointly optimized.

3. Proposed deep architecture

3.1. Feature preparation

Our basic classification model consists of 15×5 blocks of HOG and CSS features with 36 dimensions per block. The widely used version of HOG feature in [24] contains 31-dimensional feature vector, where the feature set is augmented to include both contrast sensitive and contrast insensitive features. In this implementation, 9 bins of unsigned gradient orientations, 18 bins of signed gradient orientations and 4 bins of overall gradient energy in four nearby cells are

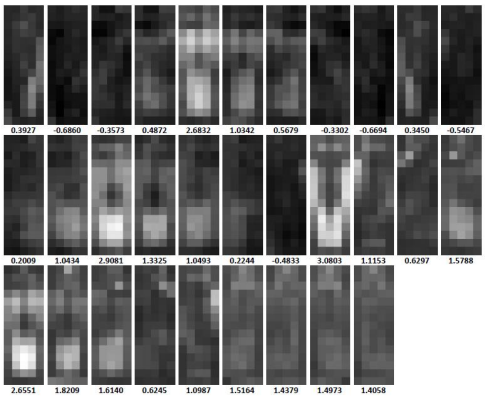

Figure 2. The discriminative power of the 31 dimensional HOG features. The 2th, 3rd, 8th, 9th, 11th, 18th dimensions are darker than other dimensions. DP_k values are shown under the maps.

used. Denote the within-class variance of the kth feature in the (i,j)th block by $V^w_{i,j,k}$, for $i = 1,\ldots,15, j = 1,\ldots 5$. Denote the between-class variance of the kth feature in the (i,j)th block by $V^b_{i,j,k}$. The discriminative power of this feature is computed by

$$DP_k = \frac{1}{15\times 5}\sum_{j=1}^{5}\sum_{i=1}^{15}\left(V^w_{i,j,k} - V^b_{i,j,k}\right). \qquad (1)$$

The discriminative power of the 31 bins is shown in Fig. 2. We discard 6 bins with the least discriminative power. Therefore, the HOG features with 25-dimensions per block are used to reduce computational load.

Color self similarity (CSS) [51] captures the pairwise statistics of spatially localized color distributions. Extensive experiments in [51, 9] show that CSS consistently improves pedestrian detection performance. It is obtained by computing the color histogram similarity between each pair of blocks. Since there are 15×5 blocks, CSS in [51] has $2,775$ dimensions. The high dimensionality of CSS makes feature extraction time-consuming. In order to reduce computation, the original $2,775$ dimensional CSS feature is modified to a 825 dimensional feature. Denote the block at position (i,j) as $B_{i,j}$ where i represents horizontal index. In our design, CSS feature for $B_{i,j}$ is $CS(B_{i,j}, B_{i+d_i,j+d_j})$ for $d_i = -2,-1,1,2, d_j = -7,\ldots,-1,1,\ldots,7$. Since CSS feature is symmetric, i.e. $CS(B_{i,j}, B_{i',j'}) = CS(B_{i',j'}, B_{i,j})$, the dimension for each block is 11.

In order to make use of contextual information in local regions, detection scores in a 3×3 spatial local region of 11 pyramids are used in our deep model. Since a pedestrian window contains $15\times 5\times 36$ features, the 3×3 detection scores for a specific pyramid is obtained by filtering the local $17\times 7\times 36$ feature pyramid with $15\times 5\times 36$ filters. Figure 3 shows an example of constructing the feature map from 3 window sizes. Contextual feature has been proved

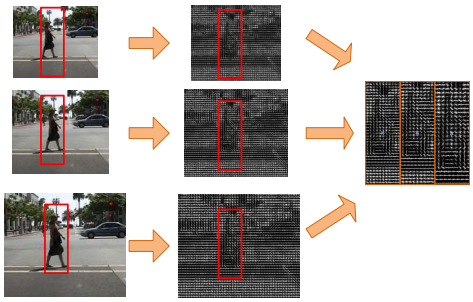

Figure 3. Construction of the feature map from three feature pyramids of different sizes. Here we only draw 3 out of 11 pyramids for the sake of page limit.

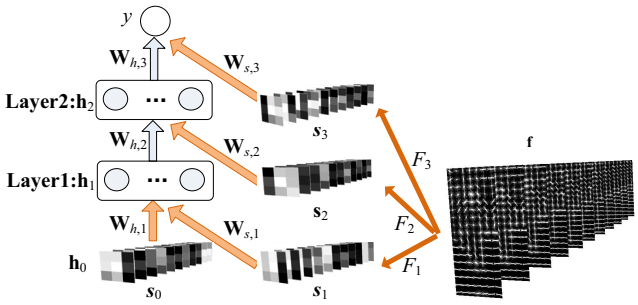

Figure 4. The proposed deep Learning Architecture. The whole architecture is built on the feature maps introduced in Section 3.1. We apply different filters F_i on the same feature map $\mathbf{f}$ and obtain different score maps $\mathbf{s}_i$. In this figure, the number of hidden layers is 2 and 3 classifiers are used. For the sake of convenience, we regard the input layer score map $\mathbf{s}_0$ as $\mathbf{h}_0$.

to be useful in [9]. The contextual score map is used in our paper, but not used in [9].

3.2. The deep architecture for inference

Figure 4 shows our proposed deep model. There are three types of nodes in the deep architecture.

- $\mathbf{f}$ is the input feature maps introduced in Section 3.1.
- $\mathbf{s}_{i+1}(\forall\ i \le L)$ is the score maps at layer i, which represent the scores of the corresponding classifier. The input score map $\mathbf{s}_0$ is obtained with linear SVM and also serves as the input layer. For the sake of convenience, we also regard $\mathbf{s}_0$ as $\mathbf{h}_0$.
- $\mathbf{h}_i(\forall\ i \le L)$ is the hidden nodes for layer i. They transfer contextual information between classifiers.

There are three types of weights connecting these nodes.

- $\mathbf{F}_{i+1}$ is the classifier at layer i used to filter the feature map and obtain the score map $\mathbf{s}_{i+1}$.

- $\mathbf{W}_{h,i}$ is the weights (transfer matrix) to connect hidden nodes $\mathbf{h}_{i-1}$ and $\mathbf{h}_i$.
- $\mathbf{W}_{s,i}$ is the weights to connect score map $\mathbf{s}_i$ and hidden nodes $\mathbf{h}_i$. As input $\mathbf{s}_0$ is regarded as $\mathbf{h}_0$, the weight matrix $\mathbf{W}_{s,0}$ between $\mathbf{h}_0$ and $\mathbf{h}_1$ is denoted by $\mathbf{W}_{h,1}$.

The input feature map $\mathbf{f}$ has 11 pyramids, each of which has $17 \times 7 \times 36$ features. This input is used by multiple classifiers, which are learned in different layers. At the inference stage, classifier $\mathbf{F}_{i+1}$ filters the feature map $\mathbf{f}$ and outputs the score map $\mathbf{s}_{i+1}$:

$$\mathbf{s}_{i+1} = \frac{1}{1+e^{-\mathbf{F}_{i+1}\otimes\mathbf{f}}}, \quad (2)$$

where $\otimes$ denotes the filtering operation. An initial score map $\mathbf{s}_0$ is obtained by filtering the feature map $\mathbf{f}$ with a classifier $\mathbf{F}_0$ that is learned by linear SVM. $\mathbf{F}_0$ is fixed and $\mathbf{s}_0$ is used as the contextual detection score information.

After score map $\mathbf{s}_i$ is obtained with Eq. 2, $\mathbf{s}_i$ and $\mathbf{h}_{i-1}$ are fully connected to the hidden nodes in $\mathbf{h}_i$ and we have

$$\mathbf{h}_1 = \frac{1}{1+e^{-(\mathbf{W}_{h,1}\mathbf{s}_0+\mathbf{W}_{s,1}\mathbf{s}_1)}}, \quad (3)$$

$$\mathbf{h}_{i+1} = \frac{1}{1+e^{-(\mathbf{W}_{h,i+1}\mathbf{h}_i+\mathbf{W}_{s,i+1}\mathbf{s}_{i+1})}}, \forall i \leq L-1 \quad (4)$$

Finally, the probability of a window containing a pedestrian is obtained as follows:

$$\mathbf{y} = \frac{1}{1+e^{-(\mathbf{W}_{h,L+1}\mathbf{h}_L+\mathbf{W}_{s,L+1}\mathbf{s}_{L+1})}}. \quad (5)$$

3.3. Stage-by-stage training of the deep model

The training procedure is summarized in Algorithm 1. It consists of two steps. The deep model is first trained by excluding additional classifiers at all the layers to reach a good initialization point. Training this simplified model avoids overfitting. Then the classifier at each layer is added one by one. At each stage t, all the existing classifiers up to layer t are jointly optimized. Each round of optimization finds a better local minimum around the good initialization point reached in the previous training stages.

- Step 1.1 (1 and 2 in Algorithm 1): the layer-by-layer unsupervised pre-training approach in [26] is used to train the hidden-to-hidden transfer matrices $\mathbf{W}_{h,i+1}$. In this step, we set $\mathbf{W}_{s,i+1} = 0$ and $\mathbf{F}_{i+1} = 0$ for $i = 0, \ldots, L$.
- Step 1.2 (3 in Algorithm 1): BP is used for fine-tuning all the $\mathbf{W}_{h,i+1}$ together, with $\mathbf{W}_{s,i+1} = 0$ and $\mathbf{F}_{i+1} = 0$.
- Step 2.1 (4 in Algorithm 1): filters $\mathbf{F}_{i+1}$, for $i = 0, \ldots, L$, are randomly initialized in order to search for extra discriminative information in the next step.

Algorithm 1: Stage-by-Stage Training

Input: Training set: $\Psi = \{\mathbf{s}_0, \mathbf{f}\}$
Filter of linear SVM: $\mathbf{W}_0$
Number of hidden layers: L
Output: Transfer matrices: $\mathbf{W}_{h,i+1}$, $\mathbf{W}_{s,i+1}$,
New filters: $\mathbf{F}_{i+1}$, $i = 0 \ldots L$.

1 Set elements in $\mathbf{W}_{s,i+1}$ and $\mathbf{F}_{i+1}$ to be 0;
2 Unsupervised pretrain all transfer matrices $\mathbf{W}_{h,i+1}$;
3 BP to fine tune all the transfer matrices $\mathbf{W}_{h,i+1}$, while keeping $\mathbf{W}_{s,i+1}$ and $\mathbf{F}_{i+1}$ as 0;
4 Randomly initialize $\mathbf{F}_{i+1}$;
5 **for** *t=0 to L* **do**
6 Use BP to update parameters from layer 0 to layer t, i.e. $\mathbf{F}_{i+1}, \mathbf{W}_{h,i+1}, \mathbf{W}_{s,i+1}, 0 \leq i \leq t$;
7 **end**
8 Output $\mathbf{W}_{h,i+1}, \mathbf{W}_{s,i+1}, \mathbf{F}_{i+1}, i = 0 \ldots L$.

- Step 2.2 (5-7 in Algorithm 1): cascaded filters $\mathbf{F}_{i+1}$ for $i = 0, \ldots, L$ are trained using BP stage-by-stage. In stage t, classifiers $\mathbf{F}_{i+1}$ and weights $\mathbf{W}_{s,i+1}$ ($\forall\, i \leq t$) up to layer t are jointly updated.

3.4. Analysis

3.4.1 Analysis on Step 1

Since $\mathbf{W}_{s,i}$ is set to 0, step 1 can be considered as training a deep belief net (DBN) [26] with input $\mathbf{s}_0$, hidden nodes $\mathbf{h}_i$, and label $\mathbf{y}$. Step 1.1 is used for unsupervised pretraining and step 1.2 is for fine tuning. In this step, the DBN uses the contextual score map obtained by linear SVM as input for classifying samples. A majority of samples would be correctly classified by this DBN.

3.4.2 Analysis on Step 2

The properties of the training strategy in step 2 are explained in four-folds. First, it simulates the soft-cascade structure. We use the log error function as the target function. Denote the label of a sample by $l \in \{1, 0\}$. Positive samples have $l = 1$ and negative samples have $l = 0$. The log error function is defined as

$$E = -l \log y - (1-l) \log (1-y). \quad (6)$$

Denote the set of parameters $\mathbf{F}_{i+1}, \mathbf{W}_{h,i+1}$ and $\mathbf{W}_{s,i+1}$ at layer i by $\Theta_i = \{\mathbf{F}_{i+1}, \mathbf{W}_{h,i+1}, \mathbf{W}_{s,i+1}\}$, where $\theta_{i,j}$ is the jth element of Θ_i. At the BP stage, the gradient for updating $\mathbf{W}_{h,i+1}$ and $\mathbf{W}_{s,i+1}$ can be formulated as follows:

$$d\theta_{i,j} = -\frac{\partial E}{\partial \theta_{i,j}} = -\frac{\partial E}{\partial y}\frac{\partial y}{\partial \theta_{i,j}} = -(y-l)\frac{\partial y}{\partial \theta_{i,j}}. \quad (7)$$

Denote the nth training sample as $\mathbf{x}_n$. Denote its estimated label as y_n. Denote its ground truth label by l_n. The parameters Θ_i in Eq. 7 are trained at step 2.2 in Algorithm 1. At the start of loop t, $\mathbf{W}_{s,i+1} = 0$. If $y_n = l_n$, i.e. a training sample has already been correctly classified, $d\theta_{i,j}$ in Eq. 7 is 0 for sample $\mathbf{x}_n$. Therefore, a correctly classified sample at the previous stage does not influence the update of parameters. For misclassified examples, the magnitude of the $d\theta_{i,j}$ is larger when the the estimation error is larger. The update of parameters is mainly influenced by misclassified examples. *Therefore, with our training strategy, a new classifier is introduced at each stage to help deal with misclassified samples while the correctly classified samples have no influence on the new classifier.* This is the core idea of our multi-stage training.

Second, $t + 1$ classifiers (i.e. their parameters Θ_i for $i = 0, \ldots, t$) are jointly optimized at stage t in step 2.2, such that these classifiers can better cooperate with each other.

Third, the deep model retains the contextual information of features and detection scores. The convolutional classifiers $\mathbf{F}_i$ use contextual features (which cover larger regions around the pedestrian with pyramids) for obtaining the score map. The score map is the second level of contextual information, and we pass the distributions of scores in local regions to the next hidden layer. Score maps in different layers jointly deal with the classification task. Traditional cascades often lose such information. When a detection window is rejected at an early cascade stage, its features and its detection scores are not available at the next stage.

Fourth, the whole training procedure helps to avoid overfitting. Deep models are so powerful that they easily overfit the training data. We pretrain transfer matrices $\mathbf{W}_{h,i}$ first in an unsupervised way, which has been proved to provide better generalization capability [19]. The supervised stage-by-stage training can be considered as adding regularization constraints to parameters, i.e. some parameters are constrained to be zeros in the early training strategies. At each stage, the whole network is initialized with a good point reached by previous training strategies and the additional filters deal with misclassified hard samples. It is important to set $\mathbf{W}_{s,t+1} = 0$ and $\mathbf{F}_{t+1} = 0$ in the previous training strategies; otherwise, it become standard BP. With standard BP, even an easy training sample can influence any classifier. Training samples will not be assigned to different classifiers according to their difficulty levels. The parameter space of the whole model is huge and it is easy to overfit.

4. Experiments

At both training and testing stages, we use the HOG and CSS features described in Section 3.1 and a linear SVM classifier to generate score maps as the input of the bottom layer. A conservative threshold is used to prune samples and to reduce the computational load. The score map in each layer is generated in a 3×3 window and we combine 11 pyramids with the maximum score aligned to be the center of score map. Our experiments show that the extra testing time required by our network is less than 10% of the SVM we used, while training time of our strategy is two times more than standard BP .

We use log-average miss rate, i.e. the average of nine FPPI rates evenly spaced in log-space in the range from 10^{-2} to 10^{0}, to indicate the overall performance as suggested in [14] and plot miss rate vs False-Positive-Per-Image(FPPI) curves using the evaluation code provided in [14].

4.1. Overall Performance

The experiments are conducted on three public datasets, the Caltech dataset [14], the ETHZ dataset [20] and the TUD-Brussels dataset [54]. We focus on the reasonable subset, i.e. images with 50-pixel or taller, unoccluded or partially occluded pedestrians. The performance of our proposed model is compared with other relevant approaches: VJ [50], Shapelet [44], PoseInv [32], ConvNet-U-MS [46], FtrMine [13], HikSvm [36], HOG [6], MultiFtr [53], Pls [45], HogLbp [52], LatSvm-V1 [22], LatSvm-V2 [24], MultiFtr+CSS [51], FPDW [11], ChnFtrs [12], DN-HOG [38] MultiFr+Motion [51], MultiResC [42], CrossTalk [10], Contextual Boost [9] whose results are published in [14]. Our model is denoted as ContDeepNet.

4.1.1 Performance on Caltech

We use the Caltech Training Dataset as training data, and test on the Caltech Testing Dataset. Figure 5 shows the experimental results. The compared approaches have used many features, such as Haar-like features, shapelet, HOG, LBP, CSS. Part based models, which are not used in this paper, are used in [24, 9]. Linear SVMs are used in [6, 52], kernel SVM is used in [36], and cascade classifiers are used in [50, 11, 12, 10, 9]. Deep models are used in [46, 38]. Both MultiResC [42] and ContextualBoost [9] use HOG+CSS as features like us. Contextual Boost uses cascaded classifiers but it does not optimize the classifiers jointly. They have log-average miss rate 48%. These two approaches have the lowest log-average miss rate among existing approaches. The log-average miss rate of our proposed method decreases to 45% with 3% improvement compared with MultiResC and Contextual Boost.

4.1.2 Performance on ETHZ

Figure 6 shows the experimental results on the ETHZ pedestrian dataset. As most approaches are trained on the INRIA training dataset and test on this dataset, our proposed deep model is also trained on the INIRA training dataset. The ConvNet-U-MS stands for the results of the convolutional network model reported in [46]. The log-average

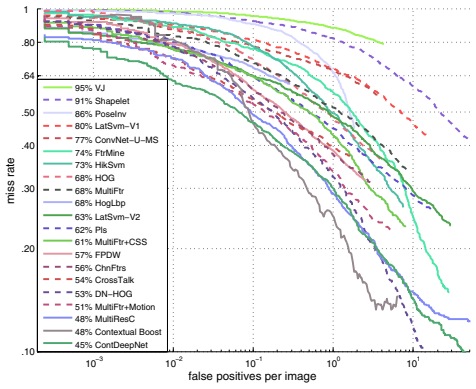

Figure 5. Comparison on the Caltech testing dataset. ContDeepNet is our proposed multi-stage contextual deep model trained on the Caltech training dataset.

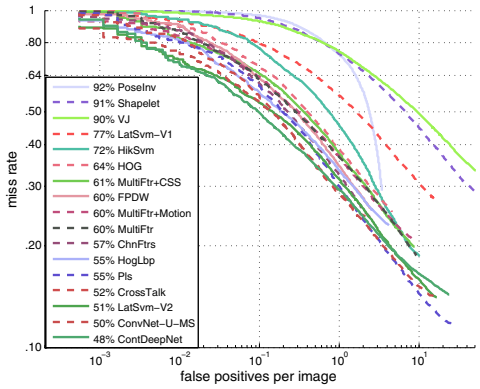

Figure 6. Comparison on the ETHZ dataset. Our ContDeepNet is trained on the INRIA training dataset.

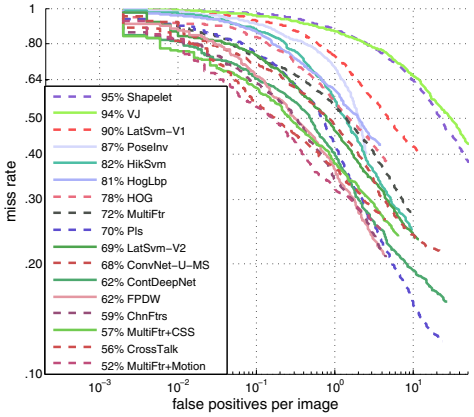

Figure 7. Comparison on the TUD dataset. Our ContDeepNet is trained on the INRIA training dataset.

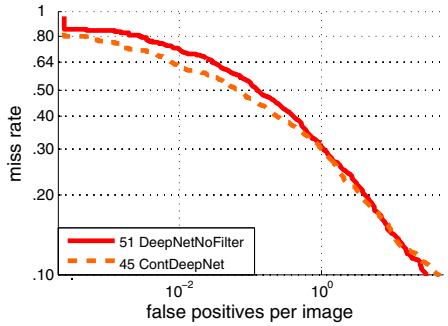

Figure 8. Comparison of different deep architectures on the Caltech Test dataset. DeepNetNoFilter is the same as ContDeepNet, except that it has no additional classifiers.

miss rate of our approach is 48% and has 2% improvement over the ConvNet-U-MS, which is the lowest among existing approaches. ConvNet-U-MS used a deep model to learn low-level features. But it does not employ contextual score maps or multi-stage classifiers.

4.1.3 Performance on TUD-Brussels

Figure 7 shows the experimental results on the TUD-Brussels pedestrian dataset. The model is trained on the INRIA training dataset. The log-average miss rate of our approach is 63%. There are a few methods outperforming ours. They have employed more features. The best performing MultiFtr+Motion [51] used motion features.

4.2. Comparsion on Architectures

In this section, we show the experimental results when different architectures are used. We compare the performance of two 3-layer deep networks. The first network, denoted by DeepNetNoFilter, has no additional classifiers. And the second network is our proposed ContDeepNet with 3 additional classifiers. Both networks are the same in other aspects, trained on the Caltech Train dataset and tested on the Caltech Test dataset.

As shown in Figure 8, the log-average miss rate decreases by 6% when including additional classifiers. Fig.9 shows the detection samples that are correctly classified by ContDeepNet but misclassified by DeepNetNoFilter. They are selected from the 300 detection samples of the two approaches with the highest detection scores. The additional classifiers help our deep model on handling hard samples. For example, the false positives of bus light, tyre, and trunk are correctly rejected and the false negatives of pedestrians with side view, blurring effect, occlusions and riding bicycles, are correctly detected.

4.3. Comparsion on Training Strategies

We design a set of experiments with the same architecture as our ContDeepNet but different training strategies. Figure 10 shows the expeimental results. The first training strategy, denoted by BP, randomly initializes all the parameters without layer-wise pretraining, then applies

Figure 9. Detection results that are missclassified by DeepNet-NoFilter but correctly classified by ContDeepNet. Results are obtained from the 300 detection samples of the two approaches with the highest detection scores.

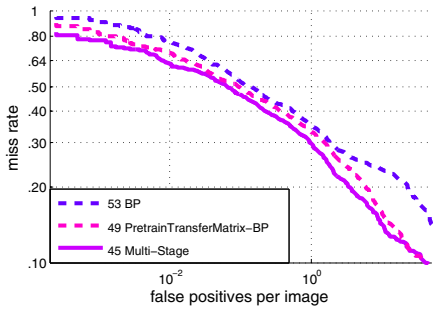

Figure 10. Results of the same architecture as our ContDeepNet but with different training strategies on the Caltech test dataset. BP means the whole network is trained with BP only. PretrainTransferMatrix-BP means the transfer matrices are unsupervised pretrained, and then all the parameters of the network are fine-tuned with BP. Multi-stage is the training strategy proposed by us.

back-propagation to simultaneously update all the transfer matrices and filters. The second algorithm, denoted by PretrainTransferMatrix-BP, adopts the method introduced in [26] to unsupervised pretrain all the transfer matrices and then uses BP to fine tune the whole network. Our proposed training strategy (Multi-Stage) has the same unsupervised pretraining procedure. However, it employs state-by-stage BP instead of the standard BP to train the whole network. The experimental results show the effectiveness of our training strategies.

5. Conclusion

In this paper, we propose a new multi-stage contextual deep model and specially designed training strategies for pedestrian detection. It simulates the cascaded classifiers. Contextual information from pyramids of feature maps and score maps propagate through the cascade. All the classifiers in the deep model are jointly trained through multiple stages of back-propagation. Overfitting is avoided through unsupervised pre-training and the designed multi-stage supervised training.

6. Acknowledgement

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] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. PAMI*, 28(12):2037–2041, 2006.

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

[3] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2005.

[4] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*. 2010.

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

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

[7] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*. 2006.

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

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

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

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

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

[13] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, pages 1–8. IEEE, 2007.

[14] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. PAMI*, 34:743–761, 2012.

[15] G. Duan, H. Ai, and S. Lao. A structural filter approach to human detection. In *ECCV*. 2010.

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

[17] M. Enzweiler and D. M. Gavrila. A mixed generative-discriminative framework for pedestrian classification. In *CVPR*, 2008.

[18] M. Enzweiler and D. M. Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Trans. PAMI*, 20:2967–2979, 2011.

[19] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *JMLR*, 11:625–660, 2010.

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

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. PAMI*, 35:1915 –1929, 2013.

[22] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*. IEEE, 2008.

[23] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[24] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32:1627–1645, 2010.

[25] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Trans. PAMI*, 26:1408–1423, 2004.

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

[27] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

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

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

[30] 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.

[31] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[32] Z. Lin and L. S. Davis. A pose-invariant descriptor for human detection and segmentation. In *ECCV*, pages 423–436. Springer, 2008.

[33] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

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

[35] P. Luo, X. Wang, and X. Tang. Pedestrian parsing via deep decompositional network. In *ICCV*, 2013.

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

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

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

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

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

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

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

[43] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. In *CVPR*, 2011.

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

[45] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *ICCV*, pages 24–31. IEEE, 2009.

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

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

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

[49] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57:137–154, 2004.

[50] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63:153–161, 2005.

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

[52] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, 2009.

[53] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *Pattern Recognition*. 2008.

[54] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *CVPR*, 2009.

[55] 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.

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

[57] Z. Zhu, P. Luo, X. Wang, and X. Tang. Deep learning identity preserving face space. In *ICCV*, 2013.