Hierarchical Feature Pooling with Structure Learning: A new method for Pedestrian Detection

Xiaoyu Wang [*] Liangliang Cao [†] Rogerio Feris [†] Ankur Data [†] Tony X. Han [‡]
[*]NEC Labs America [†] IBM T.J. Watson Research Center [‡] University of Missouri
xwang@nec-labs.com liangliang.cao, rsferis, ankurd@us.ibm.com hantx@missouri.edu

Abstract

Objects such as pedestrians exhibit large intra-class variations, posing significant challenges for visual object detection. State-of-the-art part-based models explicitly model object deformations, but are limited in their ability to handle image variations incurred by other geometric and photometric changes, such as human pose, lighting, occlusions, and large appearance variations. In this paper, we propose a novel approach which uses a spatially-biased hierarchical scheme to map features into a high-dimensional space that better represents the rich set of object appearance and local deformation variations. We propose a new algorithm to jointly learn the classification function and feature pooling in this high-dimensional space, in a structured prediction setting. Our approach achieves the best detection performance on the INRIA pedestrian dataset.

1. Introduction

Pedestrian detection is an essential problem for many emerging applications such as intelligent video surveillance and automated driver assistance. In the past decade, visual learning approaches have played a fundamental role to advance the state-of-the-art in this research area [4, 10, 6]. However, there still exists a gap between the performance obtained by current systems and the accuracy level required by real-world applications.

A major challenge in building robust visual detectors is how to deal with significant intra-class variations. As shown in Figure 1, the appearance of humans varies dramatically due to articulation, pose, clothing styles, occlusions, and lighting changes. Most existing object detection methods [4, 12, 22, 16, 17, 25] construct object models with densely extracted image patch features from sliding windows. Strong spatial constraints are usually imposed by orderly concatenating these patch features into a single feature vector, therefore limiting their ability to handle significant appearance variations.

Deformable part-based models [10] relax these spatial constraints by representing an object through a collection of parts arranged in a deformable configuration. Each part captures local appearance properties while the deformable configuration is characterized by spring-like connections between parts. In order to deal with significant appearance variations that cannot be tackled by deformable parts, the notion of multiple "components" was introduced, where each component represents a particular appearance subcategory (e.g., one component for side-view object pose and another one for frontal-view). In fact, this is one of the critical steps for performance improvement as demonstrated by S. Divvala et al. [20]. On the other hand, the computational speed decreases significantly with more components, so current implementations cannot afford too many components for handling appearance variation in object detection.

State-of-the-art image classification techniques [14, 28] employ a large set of visual *codewords* to model object appearance, generating high-dimensional feature vectors usually comprised of hundreds of thousands or even millions of dimensions with sophisticated coding approaches. These methods, however, are not suitable for traditional sliding window object detection approaches, due to efficiency issues. In [14], each descriptor is encoded based on k local dictionary words. The coding coefficients are computed using an optimization process. In a sliding window setting, the whole process would be repeated for millions of windows in one image. It would take order of hours for processing an image if we naively extended a method such as [14] to perform object detection.

In this paper, we propose an alternative method which can model object appearance with millions of dimensions, while having similar computational speed to traditional deformable part-based object detectors. Our method benefits from the recent studies in scalable vocabulary tree [18] and spatial pooling [27]. Since a single several thousand dimensional model is not enough to capture all the appearance variations of complex objects such as humans, we employ multiple models by dividing object samples into different groups or nodes using a hierarchical tree structure. Each node of the tree is analogous to a component in the above referenced approaches, as it focus on a particular appearance submanifold of pedestrian images. By leveraging this hierarchical structure, we efficiently map the original low-dimensional feature vector into a sparse, high-dimensional representation, which better models the complex appearance space of humans (see Figure 2).

Our proposed method does not impose strong spatial constraints among patch features, as we allow them to be *re-ordered* in the high-dimensional space. We augment features with their relative spatial anchor position, encouraging patches that are spatially close to each other to preserve their relative positions in the new feature space. On the other hand, patches that are far away from each other spatially are not likely to be mapped towards the same dimensions in the new feature space. This is demonstrated in our experiments to be an important reason for the success of our approach. As part of our hierarchical feature mapping scheme, we map each patch in the original feature vector to several nodes in the vocabulary tree from coarse to fine resolutions, while employing a pooling process to select the patch with the most discriminative properties for each node of the tree. Different from other fixed pooling strategies used in convolutional networks [13] or bag-of-word models [7], we *learn* the pooling operation in conjunction with the classification model, in a structured prediction setting. Our approach achieves the best detection performance on the INRIA pedestrian dataset.

The rest of this paper is organized as following: Section 2 discusses related work. Section 3 introduces our features and the proposed hierarchical feature mapping. Section 4 describes our proposed stochastic training process with pooling. Section 5 covers our experiment results. Section 6 concludes this paper.

2. Related Work

Various methods have been proposed for pedestrian detection in the past decade (see [6] for a recent survey). Nearly all modern detectors employ as features some form of histogram of oriented gradients (HOG), popularized by the work of Dalal and Triggs [4]. Combining multiple feature descriptors has also been a trend for achieving state-of-the-art results. Wojek and Schiele [26] showed how a combination of Haar-like features, shapelets, shape context, and HOG features outperforms any individual feature. Wang et al. [25] proposed to combine HOG and LBP features for improved human detection. Dollar et al [5] extended the work of Viola and Jones [24] by applying Haar-like features over multiple channels of visual data, while designing clever methods for real-time processing. To cope with articulations, part-based models have been investigated by several authors [10, 2]. In particular, the approach proposed by Felzenszwalb *et al*. [10, 9, 8] has been extensively used for object detection. Although impressive results have been achieved by the aforementioned methods, they are still limited in their ability to model the complex appearance variations present in images of object categories such as humans.

Our proposed approach follows a different research direction, motivated by winning systems in the PASCAL VOC classification challenge [29, 3] and the ImageNet classification challenge [15]. These methods map local patch features to a high-dimensional space. Zhou *et al*. [29] employed high-dimensional feature mapping through *super-vector coding* of local image descriptors. Chen *et al*. [3] generalized this idea by combining hierarchical matching with high-dimensional feature mapping. Lin [15] developed efficient algorithms to train these high-dimensional features in a million-scale image classification setting. Applying similar ideas to object detection, however, is not feasible due to the computational burden incurred by high-dimensional features. In this paper, we propose a scalable approach using a novel hierarchical feature mapping scheme coupled with feature pooling, within a structure prediction learning framework.

3. Hierarchical Feature Mapping

In this section, we will introduce how we do the feature mapping combining appearance and spatial information.

In a manner similar to most sliding window approaches, each detection window in an image is divided into N_w cells/patches, as illustrated in Figure 2a in which the example is divided into four regions. For each patch, we extract the HOG and LBP features, augmented with their relative column and row anchor position coordinates to discriminate different spatial configurations. Unlike traditional approaches which concatenate these patch features as the final feature vector, this paper proposes to map these patch features into a higher-dimensional feature representation.

Figure 2 gives an intuitive illustration on how the feature mapping is performed. We obtain a nonlinear mapping by the use of a hierarchical tree structure. The tree is created from samples of both positive and negative examples. All annotated positive examples and a random portion of negative examples are used when we build the tree structure [1]. Densely extracted patch features are fed to a recursive K-means procedure to generate the tree structure.

Denote $\vec{\mathbf{F}}_j$ as the feature vector extracted from the jth patch of dimension D_F from a given detection window, and $\vec{\mathbf{C}}_j$ as the set of tree nodes passed by a patch in which nonzero entries indicate the corresponding nodes are passed by the patch. The dimension of $\vec{\mathbf{C}}_j$

[1]To overcome multiple object scales, we resize samples to the model size

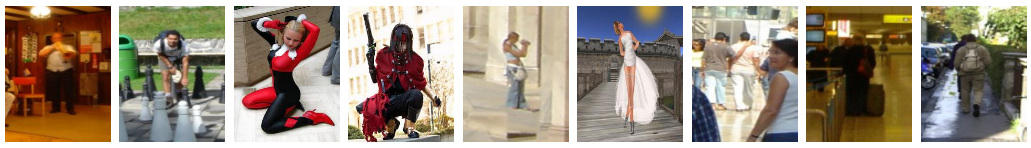


Figure 1: Large intra-class variation of human images

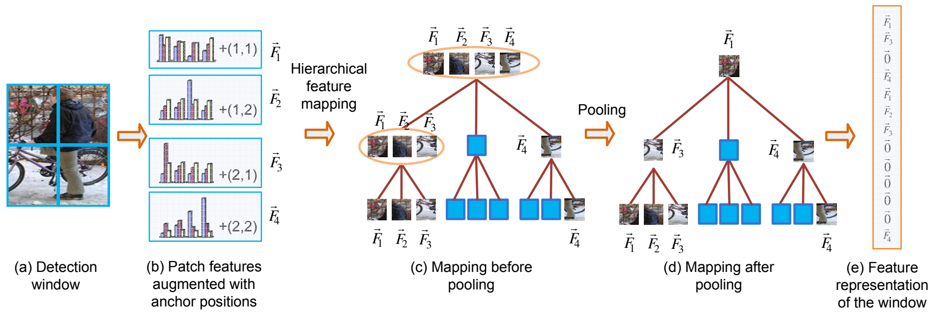

(a) Detection window
+(1,1)
+(1,2)
+(2,1)
+(2,2)
$\vec{F}_1$
$\vec{F}_2$
$\vec{F}_3$
$\vec{F}_4$
(b) Patch features augmented with anchor positions
Hierarchical feature mapping
(c) Mapping before pooling
Pooling
(d) Mapping after pooling
$\vec{F}_1$ $\vec{F}_3$ $\vec{0}$ $\vec{F}_4$ $\vec{F}_1$ $\vec{F}_2$ $\vec{F}_3$ $\vec{0}$ $\vec{0}$ $\vec{0}$ $\vec{0}$ $\vec{0}$ $\vec{F}_4$
(e) Feature representation of the window

Figure 2: Illustration of the hierarchical feature mapping. The window is divided into four patches. Many more are considered in our implementation. (a): A typical detection window divided into cells. (b):Patch features augmented with anchor positions. (c): Feature group assignment by hierarchical feature mapping. $\vec{\mathbf{F}}_1, \vec{\mathbf{F}}_2, \vec{\mathbf{F}}_3, \vec{\mathbf{F}}_4$ are features extracted from four patches respectively. (d):Feature groups after pooling. (e): Final feature representation of the detection window.

is the same as the number of tree nodes. If a node is passed by the patch $\vec{\mathbf{F}}_j$, we assign the patch to the corresponding node/group. We map the original feature $\vec{\mathbf{F}}_j$ to a new representation $\vec{\mathbf{F}}'_j = \vec{\mathbf{F}}_j \otimes \vec{\mathbf{C}}_j$ where $\otimes$ is the Kroneker product. After mapping, each $\vec{\mathbf{F}}'_j$ is a $N_T D_F$ dimensional sparse feature, where N_T is the number of tree nodes. Denote $\vec{\mathbf{M}}_j$ as the mask of $\vec{\mathbf{C}}_j$ which indicates whether the jth patch is active (i.e., whether it will be selected to be part of the final representation) for the assigned group or not. Then the resulting polished mapping is $\vec{\mathbf{F}}'_j = \vec{\mathbf{F}}_j \otimes (\vec{\mathbf{C}}_j \ \& \ \vec{\mathbf{M}}_j)$, where the dimension of $\vec{\mathbf{M}}_j$ is the same as the number of tree nodes. & is the bitwise and operation. Elements of $\{\vec{\mathbf{M}}_j\}_{j=1}^{N_w}$ satisfy $\sum_{j=1}^{N_w} m_{jk} \leq 1$ for each group index k, which means each tree node can be associated with at most one patch. Using a matrix representation M for $\{\vec{\mathbf{M}}_j\}_{j=1}^{N_w}$, the final feature representation of the detection window can be expressed as:

$$\Phi(x, y, M) = \sum_{j=1}^{N_w} \vec{\mathbf{F}}'_j = \sum_{j=1}^{N_w} \vec{\mathbf{F}}_j \otimes (\vec{\mathbf{C}}_j \ \& \ \vec{\mathbf{M}}_j) \qquad (1)$$

So, the matrix M is the feature pooling matrix which specifies which feature vector is active for each node as visualized in Figure 2d. The dimension of $\Phi(x, y, M)$ is around seven million in our experiment. We describe how to compute the mask $\vec{\mathbf{M}}_j$ in Section 4.

4. Efficient Structure SVM Training with Hierarchical Pooling

A unique property of our method is that our method does not employ any predefined pooling strategy like bag of words or spatial pyramid matching. On the contrary, we formulate the pooling step as a learning problem and let the classifier guide the pooling process. The pooling procedure is unified with feature computation, and the hierarchical structure dramatically improve the pooling speed and reduce the quantization error.

To illustrate the pooling process, we take a tree structure with three nodes as shown in Figure 3. Patch features are assigned to tree nodes hierarchically as described in the previous section. All the patches are assigned to the root node. As an example, the pooling operation for node 2 will select the feature $\vec{\mathbf{F}}_{n2} = \underset{\vec{\mathbf{F}}_j}{\arg\max} \langle \vec{\mathbf{F}}_j, \vec{\mathbf{W}}_2 \rangle, j = 1, \ldots, k$, where $\vec{\mathbf{F}}_j$ are features assigned to the tree node number 2, $\vec{\mathbf{W}}_2$ is the classification weight vector. The pooling is performed hierarchically until we reach the leaf node. Assume we have 4 patches in total and the pooled features for node 1, 2, 3 are $\vec{\mathbf{F}}_2$, $\vec{\mathbf{F}}_4$, $\vec{\mathbf{F}}_1$ respectively, then the mask M for this pooling configuration is:

$$M = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{vmatrix}. \qquad (2)$$

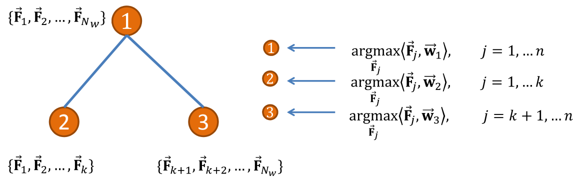

$\{\vec{\mathbf{F}}_1, \vec{\mathbf{F}}_2, \ldots, \vec{\mathbf{F}}_{N_w}\}$
1
2
3
$\{\vec{\mathbf{F}}_1, \vec{\mathbf{F}}_2, \ldots, \vec{\mathbf{F}}_k\}$
$\{\vec{\mathbf{F}}_{k+1}, \vec{\mathbf{F}}_{k+2}, \ldots, \vec{\mathbf{F}}_{N_w}\}$
$\underset{\vec{\mathbf{F}}_j}{\operatorname{argmax}} \langle \vec{\mathbf{F}}_j, \vec{\mathbf{w}}_1 \rangle, \quad j = 1, \ldots n$
$\underset{\vec{\mathbf{F}}_j}{\operatorname{argmax}} \langle \vec{\mathbf{F}}_j, \vec{\mathbf{w}}_2 \rangle, \quad j = 1, \ldots k$
$\underset{\vec{\mathbf{F}}_j}{\operatorname{argmax}} \langle \vec{\mathbf{F}}_j, \vec{\mathbf{w}}_3 \rangle, \quad j = k+1, \ldots n$

Figure 3: Illustration of Hierarchical feature pooling

The first column means $\vec{\mathbf{F}}_1$ is only assigned to node 3. The third column means $\vec{\mathbf{F}}_3$ is not associated with any nodes.

From the discussion above, we can see that with the hierarchical tree structure, patches will be first assigned to coarse groups, and then fine groups in deeper layers. The pooling is performed hierarchically for these patches with similar appearance and positions in each group. Patches far away from each other are not likely to be involved in the same pooling group, especially in the deep layer. Next we will show how the mask matrix M and the pooling procedure is learned by maximizing the classification score using structure SVMs.

In the training procedure, we have a collection of images and corresponding labeling $\{(x_i, y_i)\}_{i=1}^{N}, x_i \in \mathcal{X}, y_i \in \mathcal{Y}$. y_i is the annotation of the object in image x_i, which includes the column and row position of the object (p_1^i, p_2^i), and the scale of the object p_s^i. In the following, we denote $y\langle p\rangle$ as the bounding box of the object in the image. The feature representation of a ground truth object is jointly determined by the image x, the annotation y, and the mask M.

The goal of the object detection is to learn a discriminative function $f : \mathcal{X} \mapsto \mathcal{Y}$ which predicts the annotation y for a testing image x. Structure learning [21] has shown its outstanding performance in object detection [1, 11, 19]. Motivated by these works, we formulate the training process as a structure learning problem with latent variables. Different from previous approaches, it jointly learns the classification function and the pooling matrix M. The objective function of the regression problem is:

$$\begin{gathered}\min_{w,\xi} \tfrac{\lambda}{2}||w||^2 + \tfrac{1}{N}\sum_{i=1}^{N}\xi_i \\ s.t. \max_{M_1}(\langle\Phi(x_i, y_i, M_1), w\rangle) - \max_{M_2}(\langle\Phi(x_i, \hat{y}_i, M_2), w\rangle) \\ \geq \triangle(y_i, \hat{y}_i) - \xi_i, \xi_i > 0,\end{gathered}$$

where M_1 is the mask for the ground truth and M_2 is the mask for the candidate annotation. w is the SVM weight vector to be learned. For a given example x_i, we want the score of the ground truth annotation $\max\limits_{M_1}(\langle\Phi(x_i, y_i, M_1), w\rangle$, to be higher than $\max\limits_{M_2}(\langle\Phi(x_i, \hat{y}_i, M_2), w\rangle$, for any other annotation y. $\triangle(y_i, \hat{y}_i)$ is the loss function measuring the distance between two annotations. In our work, we compute the loss function as:

$$\triangle(y_i, y) = \begin{cases} 0 & , \text{if } y_i = y = \emptyset \\ 1 & , \text{if } y_i = \emptyset, y \neq \emptyset \\ 1 - \frac{y_i\langle p_i\rangle \cap y\langle p\rangle}{y_i\langle p_i\rangle \cup y\langle p\rangle} & , \text{if } y_i \neq \emptyset, y \neq \emptyset \end{cases}$$

where $y_i = \emptyset$ means there is no object in the image. $y_i\langle p_i\rangle \cap y\langle p\rangle$ means the intersection area of two bounding box annotations, while $y_i\langle p_i\rangle \cup y\langle p\rangle$ means the union area of two bounding boxes.

As discussed above, we map the feature representation through a tree structure to obtain $\Phi(x, y, M)$. Different patch assignment mask M used to select representative and discriminative features for groups will yield different feature representation of the detection window. Determining the mask M is essentially a feature pooling process to select the discriminative patch for each group. The training process involves the latent parameter M for each example which needs to be determined . As explained in Section 3, each column of M indicates the group assignment mask of a patch $\vec{\mathbf{F}}_j$. For groups that the patch is not a member of, the corresponding entries are 0. If a group with the index k only contains $\vec{\mathbf{F}}_j$, then $m_{jk} = 1$. For groups with multiple assignments, only the patch with the max response to the group classifier, i.e. the inner product between feature $\vec{\mathbf{F}}_j$ and the weight vector of the group classifier $\vec{\mathbf{W}}_k$, is selected and the corresponding mask is set to be 1. The final feature representation is obtained as shown in Figure 2e. If a group is not associated with a patch, we put zero as the feature vector. Feature vectors in each group are concatenated according to the group index to form the ultimate feature. Because only a small portion of groups have elements, the final feature representation is sparse. Note that the feature representation in Figure 2 is strictly tied with groups. Group classifiers are updated by these feature vectors in the training process.

Because the objective function is not differentiable, we employ the subgradient descent [19] and use a stochastic update to minimize the cost function. Our algorithm is summarized in Algorithm 1. The hierarchical pooling happens in line 5 of the algorithm.

5. Experiments

We evaluate our algorithm on the popular INRIA pedestrian dataset which contains 1832 images for training and 741 images for testing. We implement the HOG-LBP feature set proposed by [25]. In our experiment, only 6 neighborhood samples are used to compute local binary patterns which results in 33 dimensional LBP histogram features. For the hierarchical K-means tree, we set $k = 10$ with a tree depth 5, so we have around 110 thousand tree nodes in total. To align the value range of the spatial feature and the appearance feature, we normalize the anchor position features according to the model width and height. We first compare our approach against the baseline approach that consists of naive concatenation of cell features to show the advantage of our feature mapping and pooling method. Then, we compare our approach against other state-of-the-art approaches.

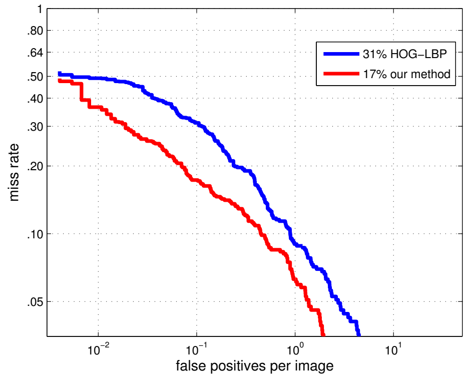
Algorithm 1: Stochastic subgradient descent Structure SVM training with feature pooling

```
  Input: training pairs $\{x_i, y_i\}_{i=1}^N, x_i \in \mathcal{X}, y_i \in \mathcal{Y}$;
         feature map $\Phi(x, y)$, loss function
         $\triangle(y, y')$, regularizer $\lambda$; number of
         iterations $T$, stepsizes $\eta_t$ for t=1,...,$T$
1 begin
2     $w \leftarrow \overrightarrow{0}$
3     for $t$=1,..., $T$ do
4         $(x_i, y_i) \leftarrow$ randomly chosen training
          example pair
5         $\hat{y}, \hat{M_1}, \hat{M_2} \leftarrow \arg\max_{y \in \mathcal{Y}} \triangle(y_i, y) +$
          $\max_{M_1}(\langle w, \Phi(x_i, y, M_1)\rangle) -$
          $\max_{M_2}(\langle w, \Phi(x_i, y_i, M_2)\rangle)$
6         $w \leftarrow$
          $w - \eta_t(w - \frac{1}{\lambda N}[\Phi(x_i, \hat{y}, \hat{M_1}) - \Phi(x_i, y_i, \hat{M_2})]$
  Output: prediction function $f(x) =$
          $\arg\max_{y \in \mathcal{Y}} \max_{M}(\langle w, \Phi(x, y, M)\rangle)$
```

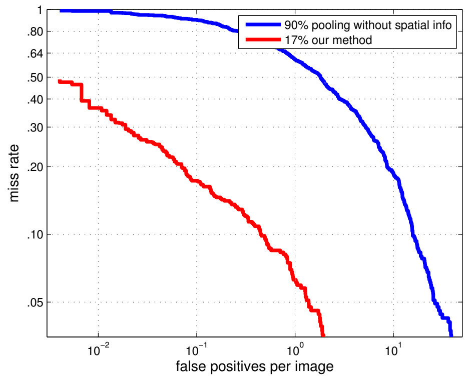
As in most of recent object detection works, we use the False Positive Per Image (FPPI)/miss rate curve to evaluate the detection performance. The detected windows with more than 50% overlap with the ground truth annotation is treated as a true detection. If an object is detected multiple times, only the one with the highest score is counted as the true detection, others are false alarms.

We first compare our approach with the baseline which aligns features strictly with respect to their spatial information. Figure 4a shows the performance comparison. The baseline approach has a miss rate 31% when FPPI=10^{-1}. Our hierarchical feature pooling with structure learning pushes it down to 17%. Note that in the baseline approach, only patches with exactly the same position are learned together. Instead, our approach takes both the appearance and patch position into account. The superior performance demonstrates that the spatial configuration relaxation significantly improve the detection performance.

An extreme of spatial configuration relaxation may just ignore all patch positions. We did another experiment in which we completely ignore the anchor positions of image patches. In this experiment setup, all the steps are exactly the same except that we did not use the spatial information when we build the tree structure, apply mapping and pooling. The blue curve in



(a)



(b)

Figure 4: Performance comparison: (a) Performance comparison between the baseline and our approach. (b) Performance with pooling without spatial information

Figure 4b shows the performance using pooling without spatial information. It demonstrates that the spatially biased pooling significantly boosted the detection performance. Without spatial information, the detector fires a lot on examples which has complex gradients. It also creates a lot of false alarms around the target object. These two experiments suggest that both the rigid template which strictly aligns features with respect to their patch anchor positions, and approaches which completely discard the spatial configuration are limited in the capability of handling large intra-class changes. Our method is kind of combination of these two. In the coarse layer of our feature mapping, the spatial positions do not play an important role in group assignment which allows for local deformations. In the deep layer, the spatial configuration difference term dominates the distance between the patch and the group. Features with relative larger spatial differences are not likely to be assigned to the same group. So spatial configurations are implicitly captured.

Figure 5 compares our method with most recent state of the art detection methods on the INRIA pedes-

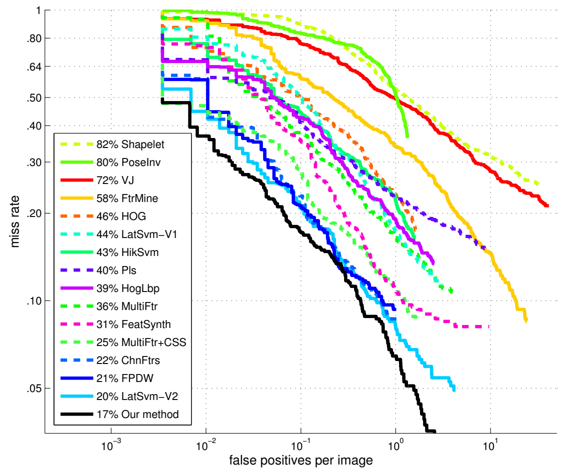


Figure 5: Performance comparison with the state of the art in the INRIA dataset

trian dataset (the performance of other detectors is obtained from [6]. We achieved the best accuracy. We achieved an Average Precision(AP) of 0.903, compared to 0.882 got by the deformable part based model. The AP obtained by the baseline approach is 0.842. The AP obtained by pooling without positions is 0.224 which is substantially lower compared to other methods.

The detection speed of our algorithm is about 7 seconds per frame with a resolution 640 by 480 which is comparable to the deformable part-based model. The proposed method can also be accelerated using cascades similar as [8].

6. Conclusion and Future Work

Combining spatially biased hierarchical feature pooling and structured regression, we proposed a framework to handle large intra-class variation in a principle way. In our approach, each image patch is mapped to several groups hierarchically based on its appearance and anchor position in the detection window. For each group with non-empty elements, the feature representation of the group is determined using the patch which has the maximum group response. The pooling procedure is naturally embedded in the training framework to achieve the best classification. We compared the proposed method with approaches with rigid feature positions as well as the approach only exploring the appearance cues. Our method obtained the best performance. We further did the performance comparison with state of the arts to demonstrate the effectiveness of the proposed approach. In our approach, the patch appearance feature and patch positions are simply concatenated. In future research, we will investigate on applying the same framework to selective bounding boxes [23]for object detection.

References

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

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

[3] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. Generalized hierarchical matching for image classification. *CVPR*, 2012.

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

[5] P. Dollar, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, 2007.

[6] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE T-PAMI*, 2011.

[7] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

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

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE T-PAMI*, 2010.

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

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

[12] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[13] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *ISCAS*, 2010.

[14] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and svm training. In *CVPR*, 2011.

[15] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and svm training. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011.

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

[17] S. Munder and D. Gavrila. An experimental study on pedestrian classification. *IEEE T-PAMI*, 2006.

[18] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[19] S. Nowozin and C. H. Lampert. Structured support vector machines. *Tutorial on "Structured Prediction and Learning in Computer Vision" in CVPR 2011.*

[20] A. A. E. Santosh K. Divvala and M. Hebert. How important are 'deformable parts' in the deformable parts model? In *ECCV*, 2012.

[21] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005.

[22] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, pages 1–8, 2007.

[23] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as Selective Search for Object Recognition. In *ICCV*, 2011.

[24] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 2001.

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

[26] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. In *DAGM symposium on Pattern Recognition*, 2008.

[27] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[28] J. Yang, K. Yu, and T. S. Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.

[29] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. *ECCV*, 2010.