Deep Learning of Scene-Specific Classifier for Pedestrian Detection

Xingyu Zeng[1], Wanli Ouyang[1], Meng Wang[1], and Xiaogang Wang[1,2]

[1] The Chinese University of Hong Kong, Shatin, Hong Kong
[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China

Abstract. The performance of a detector depends much on its training dataset and drops significantly when the detector is applied to a new scene due to the large variations between the source training dataset and the target scene. In order to bridge this appearance gap, we propose a deep model to automatically learn scene-specific features and visual patterns in static video surveillance without any manual labels from the target scene. It jointly learns a scene-specific classifier and the distribution of the target samples. Both tasks share multi-scale feature representations with both discriminative and representative power. We also propose a cluster layer in the deep model that utilizes the scene-specific visual patterns for pedestrian detection. Our specifically designed objective function not only incorporates the confidence scores of target training samples but also automatically weights the importance of source training samples by fitting the marginal distributions of target samples. It significantly improves the detection rates at 1 FPPI by 10% compared with the state-of-the-art domain adaptation methods on MIT Traffic Dataset and CUHK Square Dataset.

1 Introduction

Pedestrian detection is a challenging task of great interest in computer vision. Significant progress has been achieved in recent years [8]. However, the performance of detectors depends much on the training dataset. For example, the performance of pedestrian detectors trained on the mostly used INRIA pedestrian dataset [5] drops significantly when they are tested on the MIT Traffic dataset [33]. Fig. 1 shows that the appearance differences between the samples from the two datasets are so large that it is difficult for a detector trained on one dataset to get a satisfactory performance when being applied to the other. Manually labeling examples in all specific scenes is impractical especially when considering the huge number of cameras used nowadays. On the other hand, for applications like video surveillance, the appearance variation of a scene captured by a camera is most likely to be small. Therefore, it is practical to train a scene-specific detector by transferring knowledge from a generic dataset in order to improve the detection performance on a specific scene. Much effort has been made to develop scene-specific detectors, whose training process is aided by generic detectors for automatically collecting training samples from target scenes without manually labeling them [33,34,23,32,1,30].

Fig. 1. (a): Image from INRIA dataset. (b): Image from MIT Traffic dataset. (c): Positive samples in INRIA. (d): Negatives in INRIA. (e)-(h): Detection results of a generic detector (HOG-SVM [5] trained on INRIA) on MIT Traffic. (e): True positives. (f): True negatives. (g): False negatives. (f): False positives. Best viewed in color.

Learning scene-specific detectors can be considered as a domain adaptation problem. It involves two distinct types of data: $\mathbf{x}_s$ from the source dataset and $\mathbf{x}_t$ from the target scene, with very different distributions $p_s(\mathbf{x}_s)$ and $p_t(\mathbf{x}_t)$. The source dataset contains a large amount of labeled data while the target scene contains no or a small amount of labeled training data. The objective is to adapt the classifier trained on the source dataset to the target scene, i.e. estimating the label y_t from $\mathbf{x}_t$ using a function $y_t = f(\mathbf{x}_t)$. As an important preprocessing, we can extract features from $\mathbf{x}_t$ and have $y_t = f(\phi(\mathbf{x}_t))$, where $\phi(\mathbf{x}_t)$ is the extracted features like HOG or SIFT. We also expect that the marginal distribution $p_s(\phi(\mathbf{x}_s))$ is very different from $p_t(\phi(\mathbf{x}_t))$. Our motivation of developing deep models for scene-specific detection is three-folds.

First, instead of only adaptively adjusting the weights of generic hand-crafted features as existing domain adaptation methods [33,34,23,32], it is desirable to automatically learn scene-specific features to best capture the discriminative information of a target scene. This can be well achieved with deep learning.

Second, it is important to learn $p_t(\phi(\mathbf{x}_t))$, which is challenging when the dimensionality of $\phi(\mathbf{x}_t)$ is high, while deep models can learn $p_t(\phi(\mathbf{x}_t))$ well in a hierarchical and unsupervised way [16]. 1) In the case that the number of labeled target training samples is small, it is beneficial to jointly learn the feature representations for both $p_t(\phi(\mathbf{x}_t))$ and $f(\phi(\mathbf{x}_t))$ to avoid overfitting of $f(\phi(\mathbf{x}_t))$ since regulation is added by $p_t(\phi(\mathbf{x}_t))$. 2) $p_t(\phi(\mathbf{x}_t))$ also helps to evaluate the importance of a source sample in learning the scene-specific classifier. Some source samples, e.g. the blue sky in Fig. 1(d), do not appear in the target scene and may mislead the training. Their influence should be reduced.

Third, a target scene has scene-specific visual patterns across true and false detections, which repeatedly appear. For example, the true positives in Fig. 1 (e)

and false negatives in Fig. 1 (g) have similar patterns because pedestrians in a specific scene share similarity in viewpoints, moving modes, poses, backgrounds and pedestrian sizes when they walk on the same zebra crossing or wait for the traffic light at nearby locations. Similarly for the samples in Fig. 1 (f) and Fig. 1 (h). Therefore, it is desirable to specifically learn to capture these patterns.

These observations motivate us in developing a unified deep model that learns scene-specific visual features, the distribution of the visual features and repeated visual patterns. Our contributions are summarized below.

- Multi-scale scene-specific features are learned by the deep model.
- The deep model accomplishes both tasks of classification and reconstruction, which share feature representations with both discriminative and representative power. Since the target training samples are automatically selected and labeled with context cues, the objective function on classification encodes the confidence scores of target training samples, so that the learned deep model is robust to labeling mistakes on target training samples. In the meanwhile, an auto-encoder [16] reconstructs target training samples and models the distribution of samples in the target scene.
- With our specifically designed objective function, the influence of a training sample on learning the classifier is weighted by its probability of appearing in the target data.
- A new cluster layer is proposed in the deep model to capture the scene-specific patterns. The distribution of a sample over these patterns is used as additional features for detection.

Our innovation comes from the sights on vision problems and we well incorporate them into deep models. Compared with the state-of-the-art domain adaptation result [34], our deep learning approach has significantly improved the detection rates by 10% at 1 FPPI (False Positive Per Image) on two public datasets.

2   Related Work

Many generic human detection approaches learn features, clustered appearance mixtures, deformation and visibility using deep models [31,26,39,21,25] or part based models [9,38,24,27]. They assume that the distribution of source samples is similar to that of target samples. Our contributions aim at tackling the domain adaptation problem, where the distributions of data in the two domains vary significantly and the labeled target training samples are few or contain errors.

Many domain adaptation approaches learn features shared by source domain and target domain [14,12]. They project hand-crafted features into subspaces or manifolds, instead of learning features from raw data. Some deep models are investigated in the Unsupervised and Transfer Learning Challenge [15] and the Challenge on Learning Hierarchical Models [19]. And transfer learning using deep models has been proved to be effective in these challenges [22,13], in animal and vehicle recognition [13], and in sentiment analysis [11,3]. We are inspired by these works. However, they focus on unsupervised learning of features shared in

different domains and use the same structures and objective functions as existing deep models for general learning. We have innovation in both aspects.

A group of works on scene-specific detection [23,29,35,33,34] construct auto-labelers for automatically obtaining confident samples from the target scene to retrain the generic detector. Wang et al. [34] explore a rich set of context cues to obtain reliable target-scene samples, predict their labels and confidence scores. Their training of classifiers incorporates confidence scores and is robust to labeling errors. Our approach is in this group. The confident samples obtained by these approaches can be used as the input of our approach in learning the deep model. Another group of works [36,20] are under the co-training framework [2], in which two different classifiers on two different sets of features are trained simultaneously for the same task. An experimental comparison in [34] shows that it is easy for co-training to drift when training pedestrian detectors and its performance is much lower than the adaptive detector proposed in [34].

Samples in the source and target datasets are re-weighted differently using SVM [35,33,34] and Boosting [4,28]. However, these approaches are heuristic but do not learn the distribution of target data. Our approach learns the distribution of target samples with a deep model and uses it for re-weighting samples.

3   The Proposed Deep Model at the Testing Stage

Our full model employed at the training stage is show in Fig. 3 and Fig. 4. It accomplishes both classification and reconstruction tasks, and takes input from source and target training samples. However, at the testing stage, we only keep the parts for classification and take target samples as input. An overview of the proposed deep model for pedestrian detection in the target scene is shown in Fig. 2. This deep model contains three convolutional neural network (CNN) layers [18], three fully connected layers, the proposed cluster layer and the classification label y on whether a window contains a pedestrian or not.

The three CNN layers contain three convolutional sub-layers and three average pooling sub-layers:
- The convolutional sub-layer convolves its input data with the learned filters and then the nonlinearity function $|\tanh(x)|$ is used for each filter response. The output is the filtered data map.
- Feature maps are obtained by average pooling of the filtered data maps.
- The next convolutional layer treats feature maps as the input data and this procedure repeats for three times.

Details for convolutional sub-layers and average pooling sub-layers are as follows:
- The first convolutional sub-layer has 64 9×9×3 filters, the second has 20 2×2×64 filters and the last has 12 4×4×20 filters.
- The average pooling sub-layer down-samples the filtered data map by sub-sampling step $K \times K$ using $K \times K$ boxcar filters. $K = 4$ in the first pooling sub-layer, $K = 2$ in the second and the third sub-layer.

The fully connected layers have 2888 hidden nodes at the first layer, 2400 nodes at the second layer, and 800 nodes at the third layer. The parameters of the
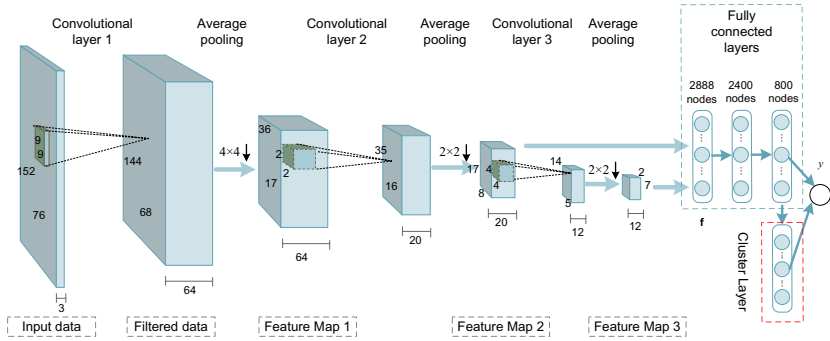
Fig. 2. Our deep model at the testing stage. There are three CNN layers, where each layer contains one convolutional sub-layer and one pooling sub-layer. The input data has three channels and is convolved with 64 9×9×3 filters, then averagely pooled within the 4 × 4 region to output the second layer. Similarly for the second layer and the third layer. The feature f is composed of the output from both the second layer and the third layer. Then the features are transferred to the fully connected layers and the cluster layer for estimating the class label y. Best viewed in color.

CNN structure are chosen by using the INRIA test set as the validation set. Details about the cluster layer is given in Section 4.5.

3.1   Input Data and Feature Preparation

We follow the approach in [26] for preparing the input data in Fig. 2. The only difference is the image size. The size of the input image is 152×76 in our implementation to have higher resolution images.

The output of all the CNN layers can be considered as features with different resolutions [31]. We concatenate the output of the second layer and the third layer in the CNN to form our features in order to use the information at different resolutions. The second layer has 20 maps of size 17×8, the third layer has 12 maps of size 7×2. Thus we obtain 2888-dimensional features, which is the **f** in Fig. 2. In this way, information at different resolutions are kept.

4   Training the Deep Model

4.1   Multi-stage Learning of the Deep Model

The overview of the stages in learning the deep model is shown in Fig. 3. It consists of the following steps:

- **(1) Obtaining Confident Target Training Samples**. Confident positive and negative training samples can be collected from the target scene using any existing approach. The method in [34] is used in our experiment. It starts with a generic detector trained on the source training set (INRIA dataset)
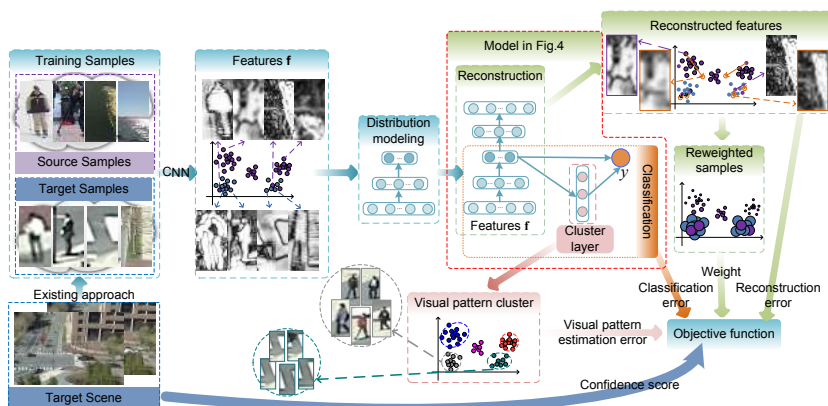
Fig. 3. Overview of our deep model. Confident samples are obtained from the target scene. Then features and their distributions in the target scene are learned. Scene-specific patterns are clustered and used for classification. Auto-encoder is used for reconstructing features and reweighting samples. The objective function is a combination of reconstruction error, visual pattern estimation error, and classification error weighted by reconstruction error and confidence score. Classification error, reconstruction error and visual pattern error are the first, second and third terms for the proposed objective function in (7) used for training the model in Fig. 4. Best viewed in color.

and automatically labels training samples from the target scene with additional context cues, such as motions, path models, and pedestrian sizes. Since automatic labeling contains errors, a score indicating the confidence on the predicated label is associated with each target training sample. Both target and source training samples are used to re-train the scene-specific detector.

- **(2) Feature Learning**. With target and source training samples, three CNN layers in Fig. 2 are used for learning discriminative features for the pedestrian detection task.
- **(3) Distribution Modeling.** The distribution of features in the target scene is learned with the deep belief net [16] using the target samples only.
- **(4) Scene-Specific Pattern Learning**. A cluster layer in our deep model is learned for capturing the scene-specific visual patterns.
- **(5) Joint Learning of Classification and Reconstruction**. Since target training samples are error prone, source samples are used to improve training. The target training samples have their classification estimation error weighted by their confidence scores in the objective function, in order to be robust to labeling mistakes. In addition to learning the discriminative information for the scene-specific classifier, an auto-encoder is included so that the deep model can learn the representative information in reconstructing the features. With a new objective function, the reconstruction error is used for reweighting the training samples. Samples better fitting the distribution of the target scene have smaller reconstruction errors and have larger influence on the objective function. At this stage, the parameters pre-trained in stage (2)-(4) are also jointly optimized with backpropagation.
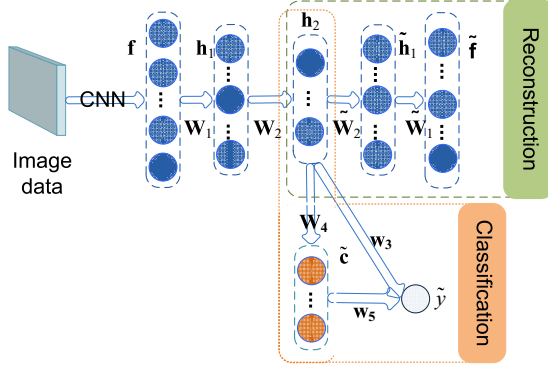
Fig. 4. Architecture of our proposed deep model at the training stage. Features **f** are extracted from the image data using three layers of CNN. In this figure, there are one input feature layer **f**, two hidden layers $\mathbf{h}_1$, $\mathbf{h}_2$, one cluster layer $\tilde{\mathbf{c}}$, one estimated classification label $\tilde{y}$, one reconstruction hidden layer $\tilde{\mathbf{h}}_1$, and one reconstructed feature layer $\tilde{\mathbf{f}}$. They are computed using Equations (1)-(6). Best viewed in color.

4.2 The Deep Model at the Training Stage

The architecture our deep model is shown in Fig. 4. The model for forward propagation is as follows:

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1^{\mathrm{T}}\mathbf{f} + \mathbf{b}_1), \tag{1}$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}_2^{\mathrm{T}}\mathbf{h}_1 + \mathbf{b}_2), \tag{2}$$

$$\tilde{\mathbf{c}} = softmax(\mathbf{W}_4^{\mathrm{T}}\mathbf{h}_2 + \mathbf{b}_4), \tag{3}$$

$$\tilde{y} = \sigma(\mathbf{w}_3^{\mathrm{T}}\mathbf{h}_2 + \mathbf{w}_5^{\mathrm{T}}\tilde{\mathbf{c}} + b_5), \tag{4}$$

$$\tilde{\mathbf{h}}_1 = \sigma(\tilde{\mathbf{W}}_2^{\mathrm{T}}\mathbf{h}_2 + \tilde{\mathbf{b}}_2), \tag{5}$$

$$\tilde{\mathbf{f}} = \sigma(\tilde{\mathbf{W}}_1^{\mathrm{T}}\mathbf{h}_i + \tilde{\mathbf{b}}_1), \tag{6}$$

where $\sigma(a) = 1/[1 + \exp(-a)]$ is the activation function.

- **f** is the feature obtained from CNN.
- $\mathbf{h}_i$ for $i = 1, \ldots, L$ denotes the vector containing hidden nodes in the ith hidden layer of the deep belief net used for capturing the shared representation of the target scene. As shown in Fig. 4, we use $L = 2$ hidden layers.
- $\tilde{\mathbf{c}}$ is the vector representing the cluster layer to be introduced in Section 4.5. Each node in this layer represents a scene-specific visual pattern.
- $\tilde{y}$ is the estimated classification label on whether a window contains a pedestrian or not.
- $\tilde{\mathbf{h}}_1$ is the hidden vector for reconstruction, with the same dimension as $\mathbf{h}_1$.
- $\tilde{\mathbf{f}}$ is the reconstructed feature vector for **f**.
- $\mathbf{W}_*$, $\mathbf{w}_*$, $\mathbf{b}_*$, $\tilde{\mathbf{W}}_*$, and $\tilde{\mathbf{b}}_*$ are the parameters to be learned.

The dimensionality of $\mathbf{h}_2$ is lower than that of $\mathbf{f}$. From $\mathbf{f}$ to $\mathbf{h}_1$ to $\mathbf{h}_2$, the features $\mathbf{f}$ are represented by low dimensional hidden nodes in $\mathbf{h}_2$. $\mathbf{h}_2$ is shared on the following two paths.

- The first path, from $\mathbf{h}_2$ to $\tilde{\mathbf{c}}$ to $\tilde{y}$, is used for estimating the classification label. This path is exactly the same as the model at the testing stage in Fig. 2. $\mathbf{h}_2$ is used for classification on this path.
- The second path, from the features $\mathbf{f}$, hidden nodes $\mathbf{h}_1$, $\mathbf{h}_2$, $\tilde{\mathbf{h}}_1$ to reconstructed features $\tilde{\mathbf{f}}$, is the auto-encoder used for reconstructing the features $\mathbf{f}$. This path is only used at the learning stage. $\mathbf{f}$ is reconstructed from the low dimensional nonlinear representation $\mathbf{h}_2$ on this path.

Denote the nth training sample with extracted feature $\mathbf{f}_n$ and label y_n as $\{\mathbf{f}_n, y_n, s_n, v_n\}$ for $n = 1, \ldots, N$, where $v_n = 1$ if $\mathbf{f}_n$ is from the target data and $v_n = 0$ if $\mathbf{f}_n$ is from the source data, s_n is the confidence score obtained by the existing approach [34] in our experiment. With the source samples and the target samples, the objective function for back-propagation (BP) learning of the deep model in Fig. 4 is as follows:

$$L = \sum_n e^{-\lambda_1 L^r(\mathbf{f}_n, \tilde{\mathbf{f}}_n)} L^c(y_n, \tilde{y}_n, s_n) + \lambda_2 v_n L^r(\mathbf{f}_n, \tilde{\mathbf{f}}_n) + v_n L_n^p, \qquad (7)$$

$$\text{where } L^r(\mathbf{f}_n, \tilde{\mathbf{f}}_n) = ||\mathbf{f}_n - \tilde{\mathbf{f}}_n||^2, \qquad (8)$$

$$L^c(y_n, \tilde{y}_n, s_n) = s_n L^E(y_n, \tilde{y}_n), \qquad (9)$$

$$L^E(y_n, \tilde{y}_n) = -y_n \log \tilde{y}_n - (1 - y_n) \log(1 - \tilde{y}_n). \qquad (10)$$

- $L^r(\mathbf{f}_n, \tilde{\mathbf{f}}_n)$ is the error of the auto-encoder in reconstructing $\mathbf{f}_n$.
- $L^E(y_n, \tilde{y}_n)$ is the error in estimating the classification label y_n, which is implemented by the cross-entropy loss.
- $L^c(y_n, \tilde{y}_n, s_n)$ is the reweighted classification error. For the source sample, $s_n = 1$ and L^E is directly used. For the target sample, the confidence score $s_n \in [0\ 1]$ is used for reweighting the classification estimation error L^E so that the classifier is robust to the labeling mistake of the confident samples.
- L_n^p is the error in estimating the visual pattern membership of the target data, which is detailed in Section 4.5.
- $\lambda_1 = 0.00025$, $\lambda_2 = 0.1$ in all our experiments.

4.3   Motivation of the Objective Function

The Objective Function for Confident Target Samples. The objective function have three requirements for target samples:
- $\mathbf{h}_2$ should be representative so that the reconstruction error on target samples of the auto-encoder is small.
- $\mathbf{h}_2$ should be discriminative so that the class label estimation error is small.
- $\mathbf{h}_2$ should be able to recognize the scene-specific visual patterns.

Therefore, $\mathbf{h}_2$ should be a compact, nonlinear representation of the representative and discriminative information in the target scene.

The Objective Function for Source Samples. Denote the source sample by $\{\mathbf{f}_s, y_s, v_s\}$. Since $v_s = 0$ in (7), the source sample does not influence the learning of the auto-encoder and the cluster layer. Denote the probability of $\mathbf{f}_s$ appearing in the target scene by $p_t(\mathbf{f}_s)$.

- If $p_t(\mathbf{f}_s)$ is very low, this sample may not appear in the scene and may mislead the training procedure. Thus the influence of $\mathbf{f}_s$ on learning the scene-specific classifier should be reduced. The objective function in (7) fits this goal. In our model, the auto-encoder is used for learning the distribution of the target data. If the auto-encoder produces high reconstruction error for a source sample, this sample does not fit the representation of the target data and $p_t(\mathbf{f}_s)$ should be low. In the extreme case, $L^r(\mathbf{f}_s, \tilde{\mathbf{f}}_s) \to \infty$ and $e^{-aL^r(\mathbf{f}_s, \tilde{\mathbf{f}}_s)} \to 0$ in (7). Thus the weighted classification loss is 0 and this sample has no influence on learning the scene-specific classifier.
- If $p_t(\mathbf{f}_s)$ is high, it should be used. In this case, the sample can be well represented by the auto-encoder and has low reconstruction error. In our objective function, $e^{-aL^r(\mathbf{f}_s, \tilde{\mathbf{f}}_s)} \approx 1$ and the classification error L^c of this sample influences the scene-specific classifier.

In this way, the source samples are weighted by how they fit the low dimensional representation of the target domain. The other purpose of L^r in (7) is to require that the low-dimensional feature representation $\mathbf{h}_2$ used for classification can also well reconstruct most target samples. The regularization avoids overfitting when the number of training samples is not large and they have errors.

4.4   Learning Features and Distribution in the Target Scene

The CNN layers in Fig. 2 are used for extracting features from images. We train these layers by using source and target samples as input and putting their labels above the third CNN layer. The cross-entropy error function in (10) and BP are used for learning CNN. In this way, the features for the pedestrian detection task are pre-trained[1].

Then the distribution of features in the target scene is learned in an unsupervised way using targe samples only. This is done by treating $\mathbf{f}$, $\mathbf{h}_1$, $\mathbf{h}_2$ as a deep belief net (DBN) [16]. The weights $\mathbf{W}_1$ and $\mathbf{W}_2$ in Fig. 4 are pre-trained using the greedy layer-wise learning algorithm in [16] while all matrices connected to the cluster layer $\tilde{\mathbf{c}}$ are fixed to zero. Many studies have shown that DBN can well learn the distribution of high-dimensional data and its low-dimensional representation. Pre-trained with DBN, auto-encoder can well reconstruct high-dimensional data from this low-dimensional representation.

4.5   Unsupervised Learning of Scene-Specific Visual Patterns

This section introduces the cluster layer in Fig. 4 for capturing scene-specific visual patterns.

---

[1] They will be fine-tuned with other parts of the deep model in the final stage using BP.
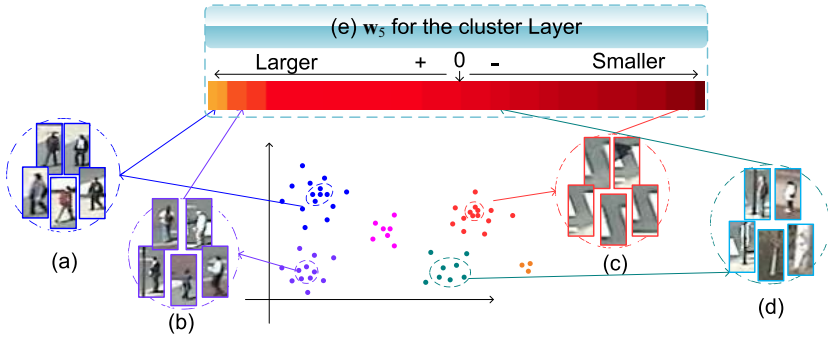
Fig. 5. Examples of scene-specific pattern clusters in the target scene and their learned weights $\mathbf{w}_5$. Pedestrians in cluster (a) are walking cross the road. Pedestrians in (b) are either waiting for green light or starting to cross the street. Samples in (c) are zebra crossings in different positions. Samples in cluster (d) contain lamp posts, trees and pedestrians. Each cluster share a similar appearance. Each cluster corresponds to a node in the cluster layer $\tilde{\mathbf{c}}$ and a weight in vector $\mathbf{w}_5$. The corresponding learned weights in $\mathbf{w}_5$ for estimating the class label $\tilde{y}$ are delineated for the patterns. $\tilde{y} = \sigma(\mathbf{w}_3^\mathrm{T}\mathbf{h}_2 + \mathbf{w}_5^\mathrm{T}\tilde{\mathbf{c}} + b_3)$. The clusters (a)(b), which mainly contain positive samples, have large positive weights in $\mathbf{w}_5$. The pattern (d), which contains mixed positive and negative samples, has its corresponding weight close to zero. Best viewed in color.

Scene-Specific Pattern Preparation. In order to capture scene-specific appearance patterns, e.g. pedestrians walking on the same road or zebra crossing, we cluster selected target samples into subsets with similar appearance. The features $\mathbf{f}$ learned from the CNN is used as the input for clustering.

We use the affinity propagation (AP) clustering method [10] to get initial clustering labels. AP fits our goal because it automatically determines the number of clusters and produces reasonable results. Fig. 5 shows some clustering results, where the visual patterns in the scene for positive and negative samples are well captured. The number of nodes in the cluster layer is set as the cluster number produced by AP. Each node in this layer corresponds to a cluster. The cluster labels of target samples are used for training the cluster layer. 51 clusters are found on the MIT Traffic dataset.

Training the Cluster Layer. The input of the nodes in the cluster layer take the combination of feature representation $\mathbf{h}_2$ with matrix $\mathbf{W}_4$. With CNN, $\mathbf{W}_1$, and $\mathbf{W}_2$ in Fig. 4 learned as introduced in Section 4.4, $\mathbf{W}_4$ in (3) is learned using the following cross-entropy error function for estimating the cluster label:

$$L_n^p = -\mathbf{c}_n^\mathrm{T} \log \tilde{\mathbf{c}}_n, \tag{11}$$

where $\mathbf{c}_n$ is the cluster label obtained by AP, $\tilde{\mathbf{c}}_n$ is the predicted cluster label. Then $\mathbf{w}_3$ and $\mathbf{w}_5$ are fine-tuned using the objective function in (7). Finally, the parameters in the CNN, the cluster layer, and the fully connect layers are fine-tuned using (7). A summary of the overall training procedure is given in Al-

---

Algorithm 1. Stage-by-Stage Training

Input: Source training set: $\Psi_s = \{\mathbf{x}_s,\ y_s\}$
confident target scene set: $\Psi_t = \{\mathbf{x}_t,\ y_t\}$
Output: CNN parameters and matrices $\mathbf{W}_i$, $\tilde{\mathbf{W}}_i$ $\forall i \leq L$, $\mathbf{w}_{L+1}$, $\mathbf{W}_{L+2}$, $\mathbf{w}_{L+3}$,
$L = 2$ in our implementation.

1 Learn scene-specific features in CNN;
2 Layer-wise unsupervised pre-training of matrices $\mathbf{W}_i$ $\forall i \leq L$;
3 BP to fine tune $\mathbf{W}_i$ $\forall i \leq L + 1$, while keeping $\mathbf{W}_{L+2}$, $\mathbf{W}_{L+3}$ as zero;
4 Cluster confident samples to obtain cluster label $\mathbf{c}_n$ for the nth sample using AP and set the number of nodes in $\mathbf{c}$ according the number of clusters obtained;
5 Fix $\mathbf{W}_i$ $\forall i \leq L$, randomly initialize $\mathbf{W}_{L+2}$, then BP to fine tune $\mathbf{W}_{L+2}$ using $\mathbf{c}_n$ as ground truth. L^p in (11) is used as the objective function ;
6 Randomly initialize $\mathbf{w}_{L+3}$. BP to fine tune $\mathbf{w}_{L+1}$ and $\mathbf{w}_{L+3}$ using the objective function in (7) ;
7 BP to fine tune all parameters using the objective function in (7) ;
8 Output parameters.

---

gorithm 1. L_n^p in (11) is used in (7) for constraining that the learned appearance pattern does not deviate far from the initial pattern found by AP clustering.

5   Experimental Results

5.1   Experimental Setting

All the experiments are conducted on the MIT Traffic dataset [33] and CUHK Square dataset [32]. The MIT Traffic dataset is a 90-minutes long video at 30 fps. 420 frames are uniformly sampled from the first 45 minutes video to train the scene-specific detector. 100 frames are uniformly sampled from the last 45 minutes video for test. The CUHK Square dataset is a 60-minutes long video. 350 frames are uniformly sampled from the first 30 minutes video for training. 100 frames uniformly sampled from the last 30 minutes video for testing. The INRIA training dataset [5] is used as the source dataset. The PASCAL criterion, i.e. the ratio of the overlap region compared to the union should be larger than 0.5, is adopted. The evaluation metric is recall rate versus false positive per image (FPPI). The same experimental setting has been used in [33,34,32].

We obtain 4262 confident positive samples, 3788 confident negative samples and their confident scores from the MIT Traffic training frames with the approach in [34]. For CUHK Square, we get 1506 positive samples and 37392 negative samples for training. They are used to train the scene-specific detector together with the source dataset. During test, for the sake of saving computation, we use a linear SVM trained on both source dataset and confident target samples to pre-scan all windows and prune candidate samples in the test images with conservative thresholds, and then apply our deep learning scene-specific detector to the remaining candidates. Compared with using SVM alone, about 50 % additional computation time is introduced. When we talk about detection rates, it is assumed that FPPI = 1.

Table 1. Comparison of detection rates with state-of-the-art generic detectors on the MIT Traffic dataset and the CUHK Square dataset. The training data for 'HOG+SVM', 'ChnFtrs', 'MultiSDP' and 'JointDeep' is the INRIA dataset.

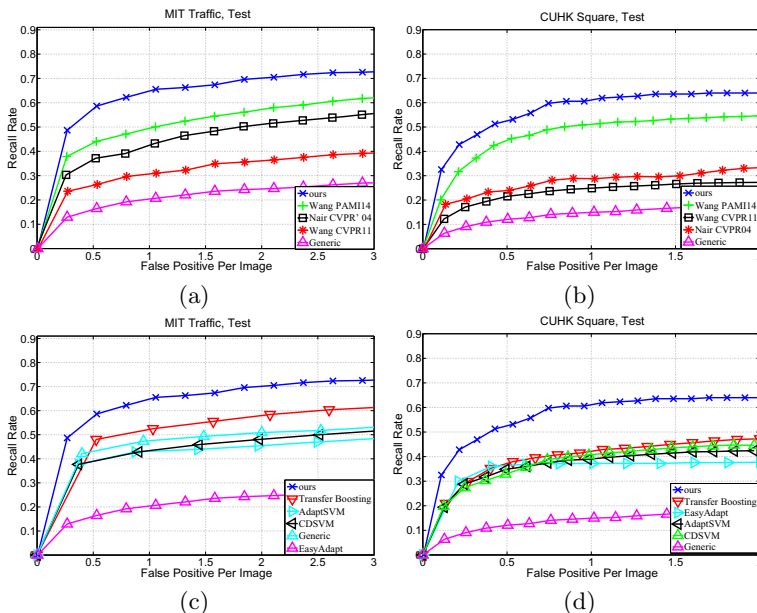|	HOG+SVM [5]	ChnFtrs [7]	MultiSDP [39]	JointDeep [26]	ours
MIT Traffic	21%	23%	23%	17%	65%
CUHK Square	15%	32%	42%	22%	62%

Fig. 6. Experimental results on the MIT Traffic dataset (left column) and the CUHK Square dataset (right column). (a) and (b): Comparison with methods requiring no manual labels from the target scene, i.e. Wang PAMI14 [34], Wang CVPR11 [33] and Nair CVPR04[23]. (c) and (d): Comparison with methods requiring manual labels on 50 frames from the target scene, i.e. Transfer Boosting [28], EasyAdapt [6], AdaptSVM [37] and CDSVM [17].

5.2 Overall Performance

We have compared our model with several state-of-the-art generic detectors [7,39,26]. The detection rates are shown in Table. 1. The training data for 'HOG+SVM', 'ChnFtrs', 'MultiSDP' and 'JointDeep' is the INRIA dataset. It is observed that the performance of the generic detectors on the MIT Traffic and CUHK Square datasets are quite poor due to the mismatch between the training data and the target scenes. They are far below the performance of our detector.

In Fig. 6(a)-(b), we compare our method with three other scene-specific approaches [23,33,34] on the two datasets. In addition to the source dataset, these approaches and ours do not require manually labeled samples from the target scene for training. 'Nair CVPR 04' in Fig. 6 represents the method in [23] which uses background subtraction to select target training samples. 'Wang CVPR11' [33] in Fig. 6 selects confident samples from the target scene by integrating

multiple context cues, such as locations, sizes, appearance and motions, and train an HOG-SVM detector. 'Wang PAMI14' [34] in Fig. 6 selects target training samples in the same way as [33] and uses a proposed Confidence-Encode SVM, which better incorporates the confidence scores, to train the scene-specific detector. Our approach obtains the target training samples in the same way as [33] and [34]. As shown in Fig. 6(a)-(b), our approach performs better than the other three methods. The detection rate of our method reaches 65% while the second best method 'Wang PAMI14' [34] is 52% on the MIT Traffic dataset. On the CUHK Square dataset, the detection rate of our method is 62% while the detection rate for the second best method in [34] is 52%.

Fig. 6(c)-(d) shows the performance of other domain adaptation approaches, including 'Transfer Boosting' [28], 'EasyAdapt' [6], 'AdaptSVM' [37], 'CDSVM' [17]. These methods all use HOG features. They make use of the source dataset and require some manually labeled target samples for training. 50 frames from the target scene are manually labeled when implementing these approaches. As shown in Fig. 6(c)-(d) , our method does not use manually labeled target samples but outperforms the second best approach ('Transfer Boosting') by 12% on MIT Traffic dataset and 16% on CUHK Square dataset.

5.3   Investigation on the Depth of CNN

In this section, we investigate the influence of the depth of the deep model on detection accuracy. All the approaches evaluated in Fig. 7 are trained on the same source and target datasets.

According to Fig. 7, 'HOG+SVM' and the deep model with one single CNN layer, named '1-layer-CNN', has similar detection performance. The '2-layer-CNN' provides 4% improvement over the '1-layer-CNN'. The '3-layer-CNN' provides 2% improvement over the '2-layer-CNN'. Therefore, the detection accuracy increases as the number of CNN layers increases from one to three. We did not observe obvious improvement by adding the fourth CNN layer. The performance increases by 2% and reaches 59% when the '3-layer-CNN' is added by two fully connected hidden layers, which is denoted by 'CNN-DBN' in Fig. 7 .

5.4   Investigation on Deep Model Design

In this section, we investiage the influence of our deep model design, i.e. the auto-encoder and the cluster layer, on the MIT Traffic dataset.

As shown in Fig. 7, the 'CNN-DBN' trained with our auto-encoder, denoted as 'CNN-DBN-AutoEncoder' in Fig. 7 , improves the detection rate by 3% compared with the 'CNN-DBN' without auto-encoder. Our final deep model with the cluster layer ('CNN-DBN-AutoEncocder-ClusterLayer') reaches detection rate 65%, which has 3% detection rate improvement compared with the deep model without the cluster layer ('CNN-DBN-AutoEncocder').

Different reweigting methods are also compared in Fig. 7. The 'CNN-DBN-Indegree' denotes the method in [34] which reweights source samples according to their indegrees from target samples. The 'CNN-DBN-AutoEncoder' denotes our
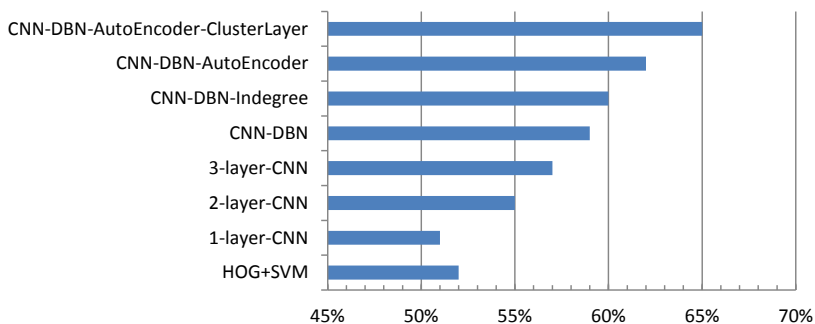
Fig. 7. Detection rates at FPPI =1 for the deep model with different number of layers and different deep model design on the MIT Traffic dataset. All the approaches, including 'HOG+SVM', are trained on the same INRIA data and the same confident target data. '1-layer-CNN' means network with only one CNN layer. '2-layer-CNN' means network with two CNN layers. '3-layer-CNN' means network with three CNN layers. 'CNN-DBN' means the model with three CNN layers and two fully connected layers. 'CNN-DBN-Indegree' means that the 'CNN-DBN' is retrained using the indegree-based reweighting method in [34]. 'CNN-DBN-AutoEncoder' is the 'CNN-DBN' retrained using our auto-encoder for reweighting samples. 'CNN-DBN-AutoEncoder-ClusterLayer' means the 'CNN-DBN-AutoEncoder' with the cluster layer. Best viewed in color.

reweighting method using the auto-encoder. Both methods are used for training the same network 'CNN-DBN'. Our reweighting method has 2% detection rate improvement compared with the indegree-based reweighting method in [34].

6   Conclusion

We propose a new deep model and a new objective function to learn scene-specific features, low dimensional representation of features and scene-specific visual patterns in static video surveillance without any manual labeling from the target scene. The new model and objective function guide learning both representative and discriminative feature representations from the target scene. Our approach is very flexible in incorporating with existing approaches that aim to obtain confident samples from the target scene.

References

1. Benfold, B., Reid, I.: Stable multi-target tracking in real-time surveillance video. In: CVPR (2011)
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: ACM COLT (1998)

3. Chen, M., Xu, Z., Weinberger, K., Sha, F.: Marginalized denoising autoencoders for domain adaptation (2012)
4. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: ICML (2007)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
6. Daumé III, H., Kumar, A., Saha, A.: Frustratingly easy semi-supervised domain adaptation. In: Proc. Workshop on Domain Adaptation for Natural Language Processing (2010)
7. Dollár, P., Tu, Z., Perona, P., Belongie, S.: Integral channel features. In: BMVC (2009)
8. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. PAMI 34(4), 743–761 (2012)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI 32(9), 1627–1645 (2010)
10. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315(5814), 972–976 (2007)
11. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: ICML (2011)
12. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: CVPR (2012)
13. Goodfellow, I.J., Courville, A., Bengio, Y.: Spike-and-slab sparse coding for unsupervised feature discovery. NIPS Workshop Challenges in Learning Hierarchical Models (2012)
14. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: ICCV (2011)
15. Guyon, I., Dror, G., Lemaire, V., Taylor, G., Aha, D.W.: Unsupervised and transfer learning challenge. In: IJCNN (2011)
16. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
17. Jiang, W., Zavesky, E., Chang, S.F., Loui, A.: Cross-domain learning methods for high-level visual concept classification. In: ICIP (2008)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS, vol. 1, p. 4 (2012)
19. Le, Q.V., Ranzato, M., Salakhutdinov, R., Ng, A., Tenenbaum, J.: Challenges in learning hierarchical models: Transfer learning and optimization. In: NIPS Workshop (2011)
20. Levin, A., Viola, P., Freund, Y.: Unsupervised improvement of visual detectors using cotraining. In: ICCV (2003)
21. Luo, P., Tian, Y., Wang, X., Tang, X.: Switchable deep network for pedestrian detection. In: CVPR (2014)
22. Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I.J., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., et al.: Unsupervised and transfer learning challenge: a deep learning approach. JMLR-Proceedings Track 27, 97–110 (2012)
23. Nair, V., Clark, J.J.: An unsupervised, online learning framework for moving object detection. In: CVPR (2004)
24. Ouyang, W., Wang, X.: Single-pedestrian detection aided by multi-pedestrian detection. In: CVPR (2013)
25. Ouyang, W., Wang, X.: A discriminative deep model for pedestrian detection with occlusion handling. In: CVPR (2012)

26. Ouyang, W., Wang, X.: Joint deep learning for pedestrian detection. In: ICCV (2013)
27. Ouyang, W., Zeng, X., Wang, X.: Modeling mutual visibility relationship in pedestrian detection. In: CVPR (2013)
28. Pang, J., Huang, Q., Yan, S., Jiang, S., Qin, L.: Transferring boosted detectors towards viewpoint and scene adaptiveness. TIP 20(5), 1388–1400 (2011)
29. Rosenberg, C., Hebert, M., Schneiderman, H.: Semi-supervised self-training of object detection models. In: WACV (2005)
30. Roth, P.M., Sternig, S., Grabner, H., Bischof, H.: Classifier grids for robust adaptive object detection. In: CVPR (2009)
31. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: CVPR (2013)
32. Wang, M., Li, W., Wang, X.: Transferring a generic pedestrian detector towards specific scenes. In: CVPR (2012)
33. Wang, M., Wang, X.: Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In: CVPR (2011)
34. Wang, X., Wang, M., Li, W.: Scene-specific pedestrian detection for static video surveillance. TPAMI 36, 361–374 (2014)
35. Wang, X., Hua, G., Han, T.X.: Detection by detections: Non-parametric detector adaptation for a video. In: CVPR (2012)
36. Wu, B., Nevatia, R.: Improving part based object detection by unsupervised, online boosting. In: CVPR (2007)
37. Yang, J., Yan, R., Hauptmann, A.G.: Cross-domain video concept detection using adaptive svms. ACM Multimedia (2007)
38. Yang, Y., Ramanan, D.: Articulated human detection with flexible mixtures of parts. PAMI 35(12), 2878–2890 (2013)
39. Zeng, X., Ouyang, W., Wang, X.: Multi-stage contextual deep learning for pedestrian detection. In: ICCV (2013)