Unsupervised Deep Domain Adaptation for Pedestrian Detection

Anonymous ECCV submission

Paper ID ***

Abstract. This paper addresses the problem of unsupervised domain adaptation on the task of pedestrian detection in crowded scenes. First, we utilize an iterative algorithm to iteratively select and auto-annotate positive pedestrian samples with high confidence as the training samples for the target domain. Meanwhile, we also reuse negative samples from the source domain to compensate for the imbalance between the amount of positive samples and negative samples. Second, based on deep network we also design an unsupervised regularizer to mitigate influence from data noise. More specifically, we transform the last full connected layer into two sub-layers– element-wise multiply layer and sum layer, add an unsupervised regularizer to further improve the domain adaptation accuracy. In experiments for pedestrian detection, the proposed method boosts the recall value by nearly 30% while precision stays almost the same. Furthermore, we perform our method on standard domain adaptation benchmarks on both supervised and unsupervised settings and also achieve state-of-the-art results.

Keywords: Unsupervised Domain Adaptation, Unsupervised Regularizer, Deep Neural Network, People Detection

1 Introduction

Deep neural network has shown great power on traditional computer vision tasks, however, the labelled dataset should be large enough to train a reliable deep model. The annotation process for the task of pedestrian detection in crowded scenes is even more resource consuming, cause we need to label concrete locations of pedestrian instances. In modern society, there are over millions of cameras deployed for surveillance. However, these surveillance situations vary in lights, background, viewpoints, camera resolutions and so on. Directly utilizing models trained on old scenes will result in poor performance on the new situations due to data distribution changes. It is also unpractical to annotate pedestrian instances for every surveillance situation.

When there are few or no labelled data in the target domain, domain adaptation helps to reduce the amount of labelled data needed. Basically, unsupervised domain adaptation aims to shift the model trained from the source domain to the target domain for which only unlabelled data are provided. Most traditional works [1–5] either learn a shared representation between source and the target

domain, or project features into a common subspace. Recently, there are also works [6–8] proposed to learn a scene-specific detector by deep architectures. However, heuristic methods are needed either on constructing feature space or re-weighting samples. Our motivation of developing a domain adaptation architecture is to reduce heuristic methods required during adaptation process.

In this paper, we propose a new approach for unsupervised deep domain adaptation for pedestrian detection. First, we utilize iterative algorithm to iteratively auto-annotate target examples with high confidence as positive pedestrian instances on the target domain. During each iteration, these auto-annotated data are regarded as training set to update the target model. However, these auto-annotated samples still have the limitations of lack of negative samples and existence of false positive samples, which will no doubt lead to exploration of predictions on non-pedestrian instances. Therefore, in order to compensate for the quantitative imbalance between positive and negative samples, we randomly sample negative instances from the source domain and mix into training set. Second, based on deep network, we further design an unsupervised regularizer to mitigate influence from data noise and avoid overfitting. More specifically, in order to have a better regularization effect during adaptation process, we propose to transform the last full connected layer of deep model into two sub-layers, element-wise multiply layer and sum layer. Thus, the unsupervised regularizer can be added on the element-wise multiply layer to adjust all weights in the deep network and gain better performance.

The contributions of our work are three folds.

- We propose an adaptation framework to learn scene-specific deep detectors for the target domains by unsupervised methodology, which adaptively select positive instances with high confidence. This can be easily deployed to various surveillance situations without any additional annotations.
- Under this framework, we combine both supervised term and unsupervised regularizer into our loss function. The unsupervised regularizer helps to reduce influence from data noise in auto-annotated data.
- More importantly, for better performance of unsupervised regularizer we propose to transform the last full connected layer of deep network into two sub-layers, element-wise layer and sum layer. Thus, all weights contained in the deep network can be adjusted under the unsupervised regularizer. To our knowledge, this is the first attempt to transform full connected layers for the purpose of domain adaptation.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 presents the details of our approach. Experimental results are shown in Section 4. Section 5 concludes the paper.

2 Relate Work

In many detection works, the generic model trained from large amount of samples on the source domain is directly utilized to detect on the target domain.

They assume that samples on the target domain are subsets of the source domain. However, when the distribution of data on target and the source domain varies largely, the performance will drop significantly. Domain adaptation aims to reduce the amount of data needed for the target domain.

Many domain adaptation works try to learn a common representation space shared between source and the target domain. Saenko et al. [1, 2] propose a both linear-transform-based technique and kernel-transform-based technique to minimize domain changes. Gopalan et al. [3] project features into Grassmann manifold instead of operating on features of raw data. Alternatively, Mesnil et al. [9] use transfer learning to obtain good representations. However, these methods have limitations since scene-specific features are not learned to boost accuracy.

Another group of works [4, 5, 10, 11] on domain adaptation is to make the distribution of source and the target domain more similar. Among these works, Maximum Mean Discrepancy (MMD) is used to as a metric to reselect samples from the source domain in order to have similar distribution as target samples. In [12], MMD is added on the last feature vector of the network as a regularization. Different from these methods, our work transforms the last full connected layer into two sub-layers on which MMD regularizer can be added. Thus our unsupervised regularizer can adjust all weights of the deep network during training.

There are also works on deep adaptation to construct scene-specific detector. Wang et al.[6] explore context cues to compute confidence, [7] learn distributions of target samples and propose a cluster layer for scene-specific visual patterns. These works re-weight auto-annotated samples for their final object function and additional context cues are needed for reliable performance. However, heuristic methods are required to re-weight samples. Alternately, Hattori el al. [8] learn scene-specific detector by generating a spatially-varying pedestrian appearance model. And Pishchulin et al. [13] use 3D shape models to generate training data. However, synthesis for domain adaptation is also costly. Compared with these methods, our approach does not include the heuristic pre-processing steps. Thus, the performances of our approach are not affected by the pre-processing steps.

3 Our Approach

In this section, we introduce our unsupervised domain adaptation architecture on the task of pedestrian detection in crowded scenes. Unsupervised domain adaptation aims to shift the model trained from the source domain to the target domain for which only unlabelled data are provided. Under the unsupervised setting, we use iterative algorithm to iteratively auto-annotate target samples and update the target model. As the auto-annotated samples may contain noises, the performances may be affected by the wrongly annotated samples. Therefore, an unsupervised regularizer is introduced to mitigate the influence from data noise on the target model. Based on the assumption that the source domain and the target domain should share the same feature space after feature extraction layers, we encode an unsupervised regularizer to make a constraint that the

distribution of data representation on the element-wise multiply layer should be similar between the source domain and the target domain.

The adaptation architecture of our approach consists of three parts – the source stream, the the target stream and the unsupervised regularizer, as shown in Fig 1. The source steam takes samples from the source domain as input, while the the target stream are trained from auto-annotated positive samples from the target domain and negative samples from the source domain. These two streams can utilize any deep detection network as their basic model, as well as their detection loss function as supervised loss functions of two streams. In our experiment, we use the detection network mentioned in Section 3.4 as the basic model. The unsupervised regularizer are integrated into the loss function of the the target stream. In the following, we will first describe our iterative algorithm which iteratively selects samples from source and the target stream models, and updates the target model accordingly (3.1). Then, we will introduce the loss function we designed for updating the target model (3.2), as well as the proposed unsupervised regularizer for improving the domain adaptation performance (3.3).

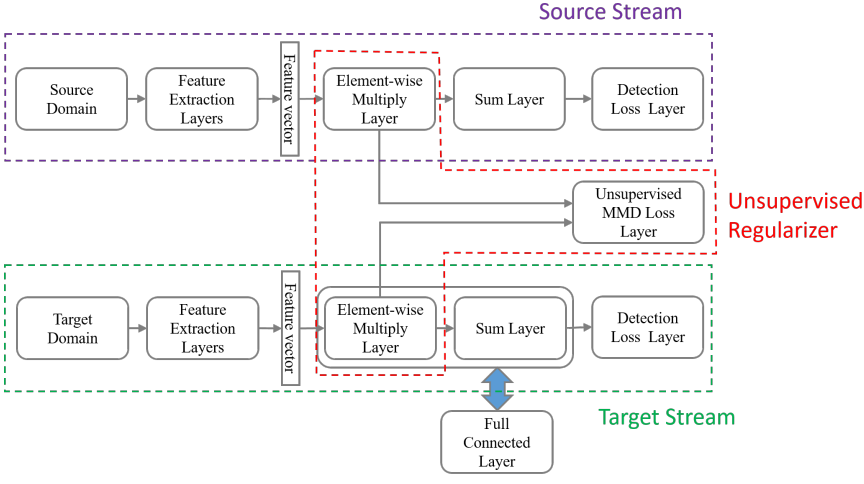

Fig. 1. The adaptation architecture consists of three parts, the source stream, the stream and an unsupervised regularizer. The last full connected layers of both source and the target stream are transformed into element-wise layer and sum layer for the purpose of the unsupervised regularizer. Best view in colors.

3.1 Iterative Algorithm

In this section, we introduce the iterative algorithm which is the training method of the the target stream of our adaptation architecture. There are two reasons to

employ iterative algorithm. First, auto-annotated data on the target domain vary for every adaptation iteration and new positive samples will be auto-annotated as training set. Compared to methods without iterative algorithm, it helps to avoid overfitting caused by lack of data. Second, unsupervised regularizer performs better with more training data as it's a distribution based regularizer.

There are two stages for iterative algorithm. The the source stream and the the target stream are separately trained at different stages. At initialization stage, the source model of the source stream are trained under supervised loss function with abundant labelled data, ($\mathbf{X}^S$,$\mathbf{Y}^S$), from the source domain. After its convergence, the weights of the source model θ^S are taken to initialize the target stream. At adaptation stage, the target model is trained from auto-annotated positive samples ($\mathbf{X}^{T,n}$,$\mathbf{Y}^{T,n}$) from the target domain and randomly-selected negative samples ($\mathbf{X}^{S,n}$,$\mathbf{Y}^{S,n}$) from the source domain under both supervised loss function and unsupervised regularizer. Since auto-annotated data are all regarded as positive samples, negative samples from the source domain are randomly selected to compensate for lack of negative instances, which are human annotated and can thus provide true negative samples. Note that we do not jointly train two streams at adaptation stage and the weights of the source model stays static which is served as a distribution reference for unsupervised regularizer at adaptation stage. The complete adaptation process is illustrated in Algorithm 1. After a predetermined iteration limit N^I is reached, we obtain our final detection model on the target domain.

Algorithm 1 Deep domain adaptation algorithm

procedure DEEP DOMAIN ADAPTATION
Train the source model on the source stream with abundant annotated data
Use well-trained the source model on the source stream to initialize the target model on the target stream as M_0
  for i = 0:N^I **do**
    M_i generate auto-annotated positive samples $\mathbf{X}^{T,n}$ of the target domain with $\mathbf{Y}^{T,n}$
    Randomly sampled negative instances $\mathbf{X}^{S,n}$ from the source domain with $\mathbf{Y}^{S,n}$
    $\mathbf{X}^n = \{\mathbf{X}^{T,n}, \mathbf{X}^{S,n}\}$
    $\mathbf{Y}^n = \{\mathbf{Y}^{T,n}, \mathbf{Y}^{S,n}\}$
    Take ($\mathbf{X}^n$, $\mathbf{Y}^n$) as training data to upgrade M_i into M_{i+1}
  end for
M_{N^I}: final model.
end procedure

Different from the source domain, we use auto-annotation tools to auto-annotate pedestrian instances with high confidence as training set on the target domain. At first iteration, we take the source model as the auto-annotation tool. For subsequent adaptation iterations, updated the target model from last adaptation iteration are utilized as the auto-annotation tool. During every adaptation iteration, the auto-annotation tool (also the target model) will be updated. Thus,

the training samples for the target domain at n^{th} adaptation iteration may differ from that at $(n+1)^{th}$ adaptation iteration.

3.2 Loss function for the the target stream

In this section, we introduce our loss function on the the target stream of adaptation architecture, which is composed of a supervised loss and an unsupervised regularizer. The supervised loss is to learn scene-specific bias of given the target domain, while the unsupervised regularizer introduced in Section 3.3 plays an important part in reducing influence from data noise as well as avoiding overfitting.

We denote training samples from the source domain as $\mathbf{X}^S = \{x_i^S\}_{i=1}^{N^S}$. For training samples on the source domain, we have corresponding annotations $\mathbf{Y}^S = \{y_i^S\}_{i=1}^{N^S}$ with $y_i^S = (b_i^S, l_i^S)$, where $b_i^S = (x, y, w, h) \in R^4$ is the bounding box location and $l_i^S \in \{0, 1\}$ is the label indicating whether x_i^S is a pedestrian instance. At n^{th} adaptation iteration, we have two set of training samples, auto-annotated positive samples from the target domain $\mathbf{X}^{T,n} = \{x_j^{T,n}\}_{j=1}^{N^{T,n}}$ and tantamount negative samples from the source domain $\mathbf{X}^{S,n} = \{x_k^{S,n}\}_{k=1}^{N^{T,n}}$. Their corresponding annotations can be denoted as $\mathbf{Y}^{T,n} = \{y_j^{T,n}\}_{j=1}^{N^{T,n}}$ and $\mathbf{Y}^{S,n} = \{y_k^{S,n}\}_{k=1}^{N^{T,n}}$ with $y_j^{T,n} = (b_j^{T,n}, l_j^{T,n} \equiv 1, c_j^{T,n})$, and $y_k^{S,n} = (b_k^{S,n}, l_k^{S,n} \equiv 0)$, respectively. $c_*^{T,n}$ is the confidence given by auto-annotation tools and N^I is the maximum number of adaptation iterations. Now we can formulate the combination of supervised loss and unsupervised regularizer as follows:

$$L(\theta^{T,n}|\mathbf{X}^{T,n}, \mathbf{Y}^{T,n}, \mathbf{X}^{S,n}, \mathbf{Y}^{S,n}, \mathbf{X}^S, \theta^S) = L_S + \alpha * L_U \tag{1}$$

$$L_S = \sum_{j=1}^{N^{T,n}} H(c_j^{T,n}) * (R(\theta^{T,n}|x_j^{T,n}, b_j^{T,n}) + C(\theta^{T,n}|x_j^{T,n}, l_j^{T,n})) + \sum_{k=1}^{N^{T,n}} (R(\theta^{T,n}|x_k^{S,n}, b_k^{S,n}) + C(\theta^{T,n}|x_k^{S,n}, l_k^{S,n})) \tag{2}$$

$$L_U = L_{EWM}(\theta^T|\mathbf{X}^T, \mathbf{X}^S, \theta^S) \tag{3}$$

where L_S is supervised loss to learn the scene-specific detector and L_U is the unsupervised regularizer part. $H(\cdot)$ is a step function in order to select positive samples with high confidence among auto-annotated data on the target domain. $R(\cdot)$ is a regression loss for bounding box location, such as norm-1 loss, and $C(\cdot)$ is a classification loss for bounding box confidence, such as cross-entropy loss. And $L_{EWM}(\cdot)$, to be introduced in Section 3.3, is the MMD-based loss added on the element-wise multiply layer for unsupervised regularization. α is the coefficient balancing the effect of supervised and unsupervised loss.

3.3 Unsupervised weights regularizer on Element-wise Multiply Layer

As mentioned before, unsupervised regularizer plays an important role in in reducing influence from data noise and avoiding overfitting. In this paper, we propose to transform the last full connected layer in order to have better effect on unsupervised regularization.

Element-wise Multiply Layer In deep neural network, the last feature vector layer are taken as an important data representation of input images. However, in this paper, we take one step further to focus on the last full connected layer which serves as an decoder to decode rich information of the last feature vector into final outputs. As the source model is trained with abundant labelled data on the source domain, the weights of the last full connected layer are also well converged. A regularizer on the last full connected layer can adjust all weights of the network compared with that on the last feature vector layer. Denote the last feature vector, weights of the last full connected layer and final outputs as $\boldsymbol{f}_{(1\times N^D)}$, $\mathbf{C}_{(N^D\times N^O)}$ and $\boldsymbol{p}_{(1\times N^O)}$. N^D, N^O are the dimension of feature vector and the dimension of output layer, respectively. Thus the operation of the full connected layer can be formulated as matrix multiply:

$$\boldsymbol{p} = \boldsymbol{f} * \mathbf{C} \tag{4}$$

where

$$p_o = \sum_d f_d * C_{d,o} \tag{5}$$

Inspired by this form, we separate the above formula into two sub-operations – element-wise multiply and sum, which can be formulated as:

$$\boldsymbol{m}_o = [f_d * C_{d,o}]_{d=1}^{N^D} \tag{6}$$

$$p_o = \boldsymbol{m}_o * \overrightarrow{\mathbf{1}} \tag{7}$$

where $\mathbf{M}_{(N^O\times N^D)} = [\boldsymbol{m}_o]$ is the intermediate results of element-wise multiply operations. $\boldsymbol{m}_o$ is a vector with N^D dimensions, which will be the object of unsupervised regularizer. Finally, we can equivalent-transform the last full connected layer between the last feature vector layer and final outputs layer into element-wise multiply layer and sum layer. The transformed element-wise multiply layer is thus the last layer with weights before output layers. Fig 2 illustrates the transformation.

Unsupervised regularizer on Element-wise Multiply Layer This section introduces our unsupervised regularizer. As stated in Section 3.1, there are false positive samples among auto-annotated data, which will mislead the network and result in worse performance. Thus, we designed an unsupervised regularizer to mitigate the influence. We have the assumption that the dimensions of

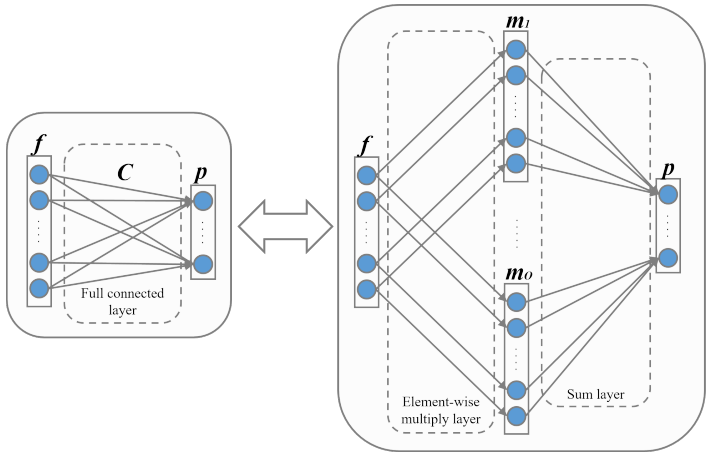

Fig. 2. Illustration of transformation of the full connected layer **C** into element-wise multiply layer and sum layer. After the transformation, the element-wise layer become the last layer which contains weights before output layer $\boldsymbol{p}$. Thus, an unsupervised regularizer can be added on $\boldsymbol{m_o}$.

element-wise multiply layer of the last full connected layer has well converged under the training of abundant source samples. Thus, when the tasks are similar, the distribution of data representations of element-wise multiply layer on the source domain and the target domain should also be similar. When we train with false positive samples, they are easier to mutate the distribution of data representations. This observation can be illustrated in Fig 3, where the center of m_o of true target samples is far closer to the center of source samples, compared to that of false target samples. Confining that the distribution of data representations between source and the target domain to be similar helps to reduce the influence caused by data noise to some extent.

To encode this similarity, we utilize MMD (maximum mean discrepancy)[4] to compute distance between distributions of element-wise multiply layer of the source domain and the target domain:

$$L_{EWM}(\theta^{T,n}|\mathbf{X}^S,\mathbf{X}^{T,n},\theta^S)=\frac{1}{N^O}\sum_{o=1}^{N^O}\|\frac{\sum_{j=1}^{N^{T,n}}(\boldsymbol{m}_o^{T,n}|x_j^{T,n})}{N^{T,n}}-\frac{\sum_{i=1}^{N^S}(\boldsymbol{m}_o^{S}|x_i^{S})}{N^{S}}\|^2 \tag{8}$$

which can also interpreted as the Euclidean distance between the center of $\boldsymbol{m}_o^{T,n}$ and $\boldsymbol{m}_o^S$ across all output dimensions. As a comparison, the MMD regularizer on feature vector layer can be formulated as:

$$L_{FV}(\theta^{T,n}|\mathbf{X}^S,\mathbf{X}^{T,n},\theta^S)=\|\frac{\sum_{j=1}^{N^{T,n}}(\boldsymbol{f}^{T,n}|x_j^{T,n})}{N^{T,n}}-\frac{\sum_{i=1}^{N^S}(\boldsymbol{f}^{S}|x_i^{S})}{N^{S}}\|^2 \tag{9}$$

where $\boldsymbol{f}$ is data of the feature vector layer in Equation 4 and $\boldsymbol{m}_o$ is the data of the element-wise multiply layer in Equation 6.

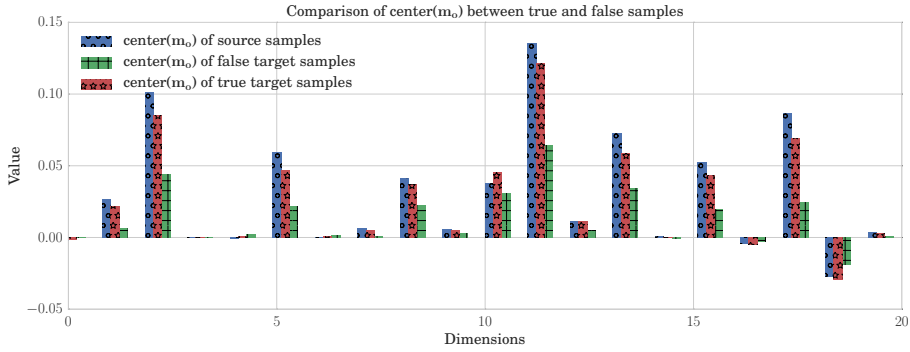

Fig. 3. Comparison of the center of m_o between true and false samples on the first 20 dimensions. The center of m_o of true target samples is far closer to the center of source samples, compared to that of false target samples. This observation supports our assumption that false instances among auto-annotated target samples tend to mutate the distribution of data representations on the element-wise multiply layer.

Since it's unpractical to get the distribution of the whole training set, while too few images cannot obtain a stable distribution for regularization. In our experiments, the $L_{EWM}(\cdot)$ loss is calculated for every batch. An example comparison of centers of $\boldsymbol{m}_o^S$ of different batches are shown in Fig 4.

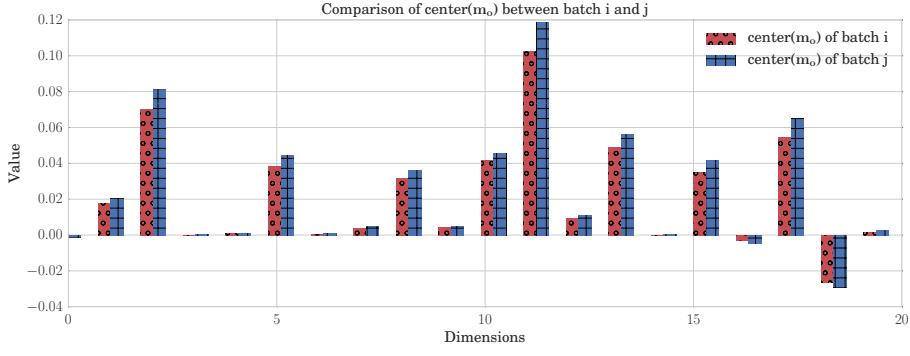

Fig. 4. Comparison of the center of $\boldsymbol{m_o}$ between two different batches on the first 20 dimensions. These two centers are close to each other, which supports our assumption that data distributions on element-wise multiply layer between source and the target domain should be similar.

3.4 Detection Network

Our generic detection model of adaptation architecture can be implemented by many deep detection models. In our experiemnt, we use the model proposed by

Russel et al. [14], which is an end to end detection network without any pre-computed region proposals needed. It consists of a GoogLeNet [15] for feature extraction and a RNN-based decoder for outputs of bounding boxes and corresponding confidences. Firstly, the GoogLeNet model encode the image into a feature map ($15 \times 20 \times 1024$) of which each 1024 dimension vector are data representation of its receptive field corresponding to a subregion of the image. Then the RNN-based layers will decode the data representation and sequentially predict 5 possible bounding boxes for each subregion by the order of their corresponding confidences. Finally, all outputs are summarized to give final detection results. However, among bounding boxes predicted as negative instances, both pedestrian and non-pedestrian instances are mixed. Therefore, we impede back-propagation of negative labels on the target domain during adaptation process.

4 Experiment Results

In this section, we introduce our experiment results on both surveillance applications and the standard domain adaptation dataset. Our motivation for unsupervised domain adaptation method is easier deployment of deep detection model on surveillance situations. Thus, we firstly evaluate our approach on video surveillance. Then we employ our approach to standard domain adaptation benchmarks on both supervised and unsupervised settings to demonstrate the effectiveness of our method.

4.1 Domain Adaptation on Crowd Dataset

Dataset and evaluation metrics To show the effectiveness of our domain adaptation approach for pedestrian detection, we collected a dataset[1] consisting of 3 target scenes for the target domain. These three scenes contain 1308, 1213 and 331 unlabelled images. For each scene, 100 images are annotated for evaluation. Instead of labelling the whole body of a person, we labels the head of a person as bounding box during training. The motivation for labelling only pedestrian heads comes from detection of indoor pedestrian or in crowded scenes, where the body of a person may be invisible. The dataset for the source domain are Brainwash Dataset[14]. Brainwash Dataset consists of over 11917 images from 3 crowded scenes.

Our evaluation metrics for detection uses the protocol defined in PASCAL VOC [16]. To judge a predicted bounding box whether correctly matches a ground truth bounding box, their intersection over their union must exceed 50%. And Multiple detections of the same ground truth bounding box are regarded as one correct prediction. For overall performance evaluation, the F1 score $F1 = 2 * precision * recall/(precision + recall)$ are utilized. Higher F1 score means better performance. At the same time, the precision-recall curve are also plotted.

[1] Our dataset will be made available soon.

Experimental settings We use deep learning framework Caffe [17] as the adaptation architecture of our approach. During the adaptation, we set learning rate as 0.01 and momentum as 0.5. At initialization stage, GoogLeNet weights are firstly used to initialize the source model of the source stream, while parameters in RNN layers are randomly initialized from a uniform distribution. For each iteration, 100 auto-annotated images from the target domain and 100 annotated images from the source domain are alternatively used for training. The outputs of our detection network include bounding box locations and corresponding confidence, thus there are two full connected layers between the last feature vector layer and the final outputs. In our experiments, when unsupervised regularizer on the element-wise multiply layer predicting box confidence are added already, unsupervised regularizer on element-wise multiply layer predicting bounding box locations have little performance improvement. Experiments on 3 target scenes are executed separately.

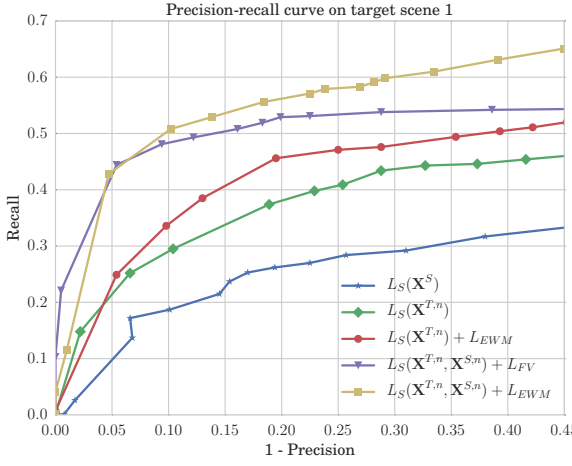

Fig. 5. Precision-recall curve of 5 comparison methods on target scene 1.

Comparison with different methods To demonstrate the effectiveness of our approach, 5 methods are compared among which method $L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{EWM}$ is our final approach:

$L_S(\mathbf{X}^S)$ Source model only trained on the source domain.

$L_S(\mathbf{X}^{T,n})$ Only auto-labeled samples on the target domain are used for training, and without any unsupervised regularizer.

$L_S(\mathbf{X}^{T,n}) + L_{EWM}$ Only auto-labeled samples on the target domain are used for training, with unsupervised MMD regularizer added on last element-wise multiply layer.

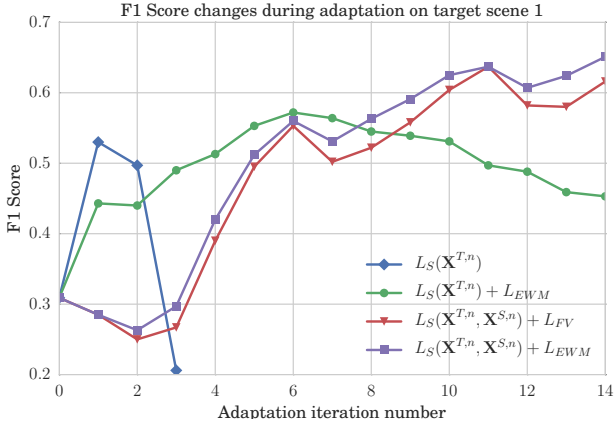

Fig. 6. F1 score changes of 5 comparison methods during adaptation on target scene 1

$L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{FV}$**[12]** Both auto-labeled images from the target domain and labeled images from the source domain are alternately sampled for training, with unsupervised MMD regularizer added on last feature vector layer.

$L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{EWM}$ Both auto-labeled images from the target domain and labeled images from the source domain are alternately sampled for training, with unsupervised MMD regularizer added on last element-wise multiply layer.

Fig 5 plots the precision-recall curve of the above comparison methods in target scene 1. Also, the F1 score changes of every adaptation iteration are also depicted in Fig 6. Table 1 gives concrete precision and recall value of the 5 comparison methods on three target scenes when the F1 scores are at their highest. Examples of adaptation results are shown in Fig 7.

Table 1. Detection results of 5 compared methods on 3 target scenes

	Scene 1			Scene 2			Scene 3		
	1-Pr	Re	F1	1-Pr	Re	F1	1-Pr	Re	F1
$L_S(\mathbf{X}^S)$	0.101	0.187	0.309	0.015	0.683	0.807	0.035	0.412	0.577
$L_S(\mathbf{X}^{T,n})$	0.245	0.408	0.530	0.632	0.905	0.524	0.176	0.778	0.800
$L_S(\mathbf{X}^{T,n}) + L_{EWM}$	0.284	0.476	0.572	0.012	0.837	**0.906**	0.078	0.653	0.764
$L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{FV}$[12]	0.109	0.496	0.637	0.002	0.721	0.838	0.044	0.611	0.746
$L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{EWM}$	0.140	0.530	**0.656**	0.006	0.811	0.893	0.097	0.778	**0.836**

Performance evaluation From the Table 1, we have the following observations:

- Compared to method $L_S(\mathbf{X}^S)$, the recall values of other methods, which all utilize iterative algorithm for training, are explicitly larger. This implies the effectiveness of our iterative algorithm on boosting recall.
- The average F1 score of $L_S(\mathbf{X}^{T,n}) + L_{EWM}$ are larger than that of method $L_S(\mathbf{X}^{T,n})$. Also, the average (1-precision) of $L_S(\mathbf{X}^{T,n})+L_{EWM}$ is far smaller. Their difference in whether the unsupervised regularizer is added into loss function demonstrates that our unsupervised regularizer can mitigate the influence of data noise and thus boost F1 score.
- Compared to method $L_S(\mathbf{X}^{T,n}) + L_{EWM}$, the average F1 score of method $L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{EWM}$ is higher. This demonstrate the effectiveness of negative source samples added into the training set during adaptation process.
- Compared to method $L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{FV}$, the recall values of method $L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n})+L_{EWM}$ are further increased. This shows that unsupervised regularizer added on the element-wise layer will provide better regularizer effect compared to that on the feature vector layer.
- Our final method $L_S(\mathbf{X}^{T,n}, \mathbf{X}^{S,n}) + L_{EWM}$ achieves best results on target scene 1 and target scene 2. The performance on target scene 2 are rather close to the best result, which may result from large discrepancy of background between source and the target domain.

4.2 Domain Adaptation on Standard Classification Benchmark

In order to further demonstrate the effectiveness and generalization of our adaptation architecture, we test our method on standard domain adaptation benchmark Office dataset[1].

Office dataset The Office dataset comprises 31 categories of objects from 3 domains (Amazon, DSLR, Webcam). Example images are depicted in Fig. 8. As Amazon domain contains 2817 labelled images, which is the largest, we take it as the source domain and Webcam domain as the target domain. We follow the standard protocol for both supervised and unsupervised settings. Specifically, for supervised domain adaptation, we use 20 randomly sampled images with labels for each category as training data for Amazon domain. When evaluate on unsupervised domain adaptation, 3 labelled images from the target domain are additional selected for each class. For both settings, the rest of images on the target domain are used for evaluation.

Experimental settings and network design On supervised setting, we reused the architecture in pedestrian detection. We utilize AlexNet [18] as the generic model of both streams. Firstly, we train the source model on the source stream with provided training data from Amazon domain. Then iterative algorithm mentioned in Sec 3.1 are utilized for adaptation. The difference is, besides auto-labelled images on the target domain 3 human labelled images for each

Fig. 7. Example results of 5 comparison methods on 3 target scenes.

Fig. 8. Example images on Office dataset.

class are also included as training data for the target model. For each iteration, 100 images are randomly sampled from training data. The unsupervised MMD regularizer is added on the element-wise multiply layer transformed from the last full connect layer of the target model. We set the coefficient value α as 0.05 on Eq 3.

We use the same experimental setting for our unsupervised adaptation, except that at adaptation stage, no human labelled images can be added into the training set.

Performance evaluation In Table 2, we compare our approach with other seven recently published works in both supervised and unsupervised settings. The outstanding performance on both settings confirms the effectiveness of our iterative algorithm and MMD regularizer on the element-wise multiply layer transformed from the last full connect layer.

Table 2. Multi-class accuracy evaluation on Office dataset with supervised and unsupervised settings.

	A → W	
	Supervised	Unsupervised
GFK(PLS,PCA)[19]	46.4	15.0
SA [20]	45.0	15.3
DA-NBNN [21]	52.8	23.3
DLID [22]	51.9	26.1
DeCAF$_6$S [23]	80.7	52.2
DaNN [11]	53.6	35.0
DDC[12]	84.1	59.4
Ours	**85.4**	**69.3**

5 Conclusions

In this paper, we introduce an adaptation architecture to learn scene-specific deep detectors for the target domains. Firstly, iterative algorithm is utilized to iteratively auto-annotate target samples and update the target model. As auto-annotated data are lack of negative samples and contain data noise, we randomly sample negative instances from the source domain. At the same time, an unsupervised regularizer is also designed to mitigate influence from data noise. More importantly, we propose to transform the last full connected layer for better regularizer effect.

Experiments on both surveillance situations and standard domain adaptation benchmarks show the effectiveness of our architecture.

References

1. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer Vision–ECCV 2010. Springer (2010) 213–226
2. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 1785–1792
3. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 999–1006
4. Huang, J., Gretton, A., Borgwardt, K.M., Schölkopf, B., Smola, A.J.: Correcting sample selection bias by unlabeled data. In: Advances in neural information processing systems. (2006) 601–608
5. Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B.: Covariate shift by kernel mean matching. Dataset shift in machine learning **3**(4) (2009) 5
6. Wang, X., Wang, M., Li, W.: Scene-specific pedestrian detection for static video surveillance. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36**(2) (2014) 361–374
7. Zeng, X., Ouyang, W., Wang, M., Wang, X.: Deep learning of scene-specific classifier for pedestrian detection. In: Computer Vision–ECCV 2014. Springer (2014) 472–487
8. Hattori, H., Naresh Boddeti, V., Kitani, K.M., Kanade, T.: Learning scene-specific pedestrian detectors without real data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3819–3827
9. 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. ICML Unsupervised and Transfer Learning **27** (2012) 97–110
10. Gong, B., Grauman, K., Sha, F.: Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In: Proceedings of The 30th International Conference on Machine Learning. (2013) 222–230
11. Ghifary, M., Kleijn, W.B., Zhang, M.: Domain adaptive neural networks for object recognition. In: PRICAI 2014: Trends in Artificial Intelligence. Springer (2014) 898–904
12. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
13. Pishchulin, L., Jain, A., Wojek, C., Andriluka, M., Thormählen, T., Schiele, B.: Learning people detection models from few training samples. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 1473–1480
14. Stewart, R., Andriluka, M.: End-to-end people detection in crowded scenes. arXiv preprint arXiv:1506.04878 (2015)
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
16. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1) (2015) 98–136

17. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
19. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2066–2073
20. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 2960–2967
21. Tommasi, T., Caputo, B.: Frustratingly easy nbnn domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 897–904
22. Chopra, S., Balakrishnan, S., Gopalan, R.: Dlid: Deep learning for domain adaptation by interpolating between domains. In: ICML workshop on challenges in representation learning. Volume 2. (2013)
23. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)