
Proceedings Paper:
Htike, KK and Hogg, D (2014) Unsupervised detector adaptation by joint dataset feature learning. In: Chmielewski, LJ, Kozera, R, Shin, B-S and Wojciechowski, K, (eds.) Computer Vision and Graphics International Conference, ICCVG 2014, Proceedings. International Conference, ICCVG, 15-17 Sep 2014, Warsaw. Springer Verlag , 270 - 277. ISBN 978-3-319-11331-9

Unsupervised Detector Adaptation by Joint Dataset Feature Learning

Kyaw Kyaw Htike and David Hogg

School of Computing, University of Leeds
Leeds, UK
{sckkh, d.c.hogg}@leeds.ac.uk

Abstract. Object detection is an important step in automated scene understanding. Training state-of-the-art object detectors typically require manual annotation of training data which can be labor-intensive. In this paper, we propose a novel algorithm to automatically adapt a pedestrian detector trained on a generic image dataset to a video in an unsupervised way using joint dataset deep feature learning. Our approach does not require any background subtraction or tracking in the video. Experiments on two challenging video datasets show that our algorithm is effective and outperforms the state-of-the-art approach.

1 Introduction

Object detection has received a lot of attention in the field of Computer Vision. Pedestrians are one of the most common object categories in natural scenes. Most state-of-the-art pedestrian detectors are trained in a supervised fashion using large publicly available generic datasets [1, 2].

However, it has been recently shown that every dataset has an inherent bias [11]. This implies that a pedestrian detector that has been specifically trained for (and is tuned to) a specific scene would do better than a generic detector for *that* scene. In fact, it has been shown that generic detectors often exhibit unsatisfactory performance when applied to scenes that differ from the original training data in some ways (such as image resolution, camera angle, illumination conditions and image compression effects) [2].

We tackle this problem by formulating a novel *unsupervised domain adaptation* framework that starts with a readily available *generic* image dataset and automatically adapt it to a target video (of a particular scene) without requiring any annotation in the target scene, thereby generating a *scene-specific* pedestrian detector.

Domain adaptation for object detectors is a relatively new area. Most state-of-the-art research use some variations of an iterative self-training algorithm [12, 7]. Unfortunately, self-training carries the risk of classifier *drifting*. In this paper, we investigate a different approach: domain adaptation purely by exploiting the *manifold property* of data.

High dimensional visual data usually exist in a nonlinear manifold that has a much smaller number of dimensions than the original data. This manifold can be

learnt using unsupervised approaches. The most relevant works to our research in this area are [6, 5]. Gopalan *et al.* [6] propose building intermediate representations between source and target domains by using geodesic flows. However, their approach requires sampling a finite number of subspaces and tuning many parameters such as the number of intermediate representations.

Gong *et al.* [5] improved on [6] by giving a kernel version of [6]. However both [6, 5] are dealing with only image data for both source and target domains and *not* videos. For videos, unique challenges are present such as the largely imbalanced nature of positive and negative data. Moreover, their approach does *not* learn deep representations required for manifolds that are highly non-linear.

In this paper we propose, using state-of-the-art deep learning, to learn the nonlinear manifold spanned by the union of the source image dataset and the sampled data of the target video. The intuition is that by learning a representation in that manifold and training a classifier on data in that representation, the resulting detector would generalize well for the target scene. This can then be used as a scene-specific detector.

Contributions. We make the following novel contributions:

1. An algorithm that adapts a pedestrian detector from an *image* dataset to a *video* using only the manifold assumption.
2. An application of state-of-the-art deep feature learning for detector adaptation in *videos* and showing its effectiveness. Furthermore, instead of starting with raw pixel values (as in standard deep learning), our approach takes as input, features such as Histogram of Oriented Gradients (HOGs) [1].
3. For videos, due to huge class imbalance, random sampling of data will result in almost all samples to be from non-pedestrian class. We propose a simple and effective biased sampling approach to minimize this problem.
4. A technique to automatically set the deep network structure with no tuning.
5. The integration of all of the above components into a system.

2 Proposed Approach

2.1 Overview

The overview of the algorithm is illustrated in Fig. 1. The algorithm is made up of two stages: (1) unsupervised deep feature learning (no supervision labels used) and (2) non-linear projection and classifier training (with labelled data). We use HOGs as the base features (before learning higher non-linear representations). In Fig. 1, we have omitted the HOG feature extraction for clarity. Our algorithm can work with any type of base features and classifier combination. However, for simplicity, we use HOGs and a linear Support Vector Machine (SVM) respectively.

Let the generic pedestrian dataset $\mathcal{G} = \{G_{\text{pos}}, G_{\text{neg}}\}$ be a set of fixed-sized pedestrian and non-pedestrian patches, $G_{\text{pos}} = \{p_1^+, p_2^+, \ldots, p_{N_1}^+\}$ and $G_{\text{neg}} = \{p_1^-, p_2^-, \ldots, p_{N_2}^-\}$ respectively where N_1 is number of pedestrian patches and

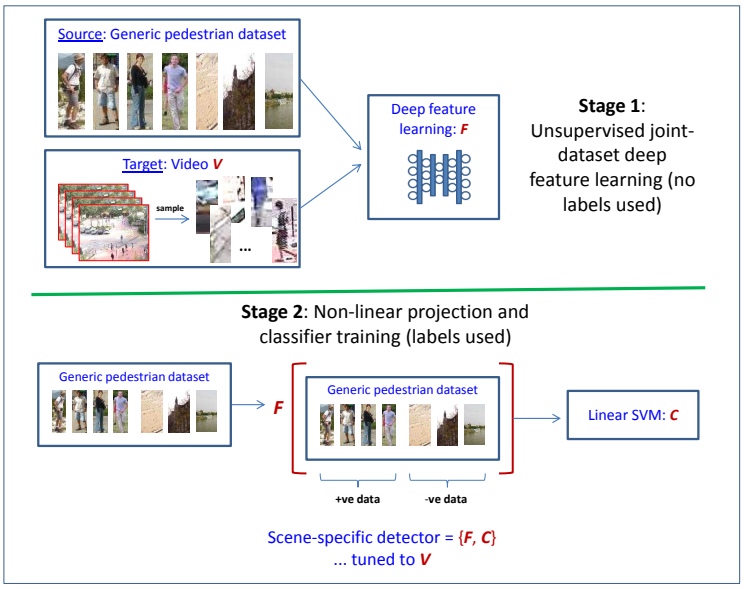

Fig. 1. Overview of the proposed algorithm.

N_2 is the number of non-pedestrian patches. Let the target video be $\mathcal{V} = \{I_1, I_2, \ldots, I_M\}$ where M is the number of frames in $\mathcal{V}$. Furthermore, let the $\mathcal{H}$ be the function to extract base features on a given image patch.

Algorithm 1 describes the detector adaptation process. The patches obtained from the generic dataset $\mathcal{G}$ are combined with the patches sampled from the target video $\mathcal{V}$. The sampling technique is detailed in Algorithm 2. After obtaining all the patches $\mathbf{D}_{\text{patches}}$, we extract HOG features from each of them, producing a feature vector for each patch. These feature vectors are input to the deep learning algorithm described in Algorithm 3. The deep learning algorithm produces, as output, a function $\mathcal{H}$ which takes in a HOG feature vector and produces a feature vector of much smaller dimension by projecting the HOG feature vector onto the learnt manifold. We then project all the HOG features of the positive and negative data of the generic dataset into this space and train a linear SVM.

Algorithm 1 Detector adaptation overview

Input: $\mathcal{G}$, $\mathcal{V}$, $\mathcal{H}$
Output: Scene-specific detector = $\{\mathcal{F}, \mathcal{C}\}$
1: $\mathbf{D}_{\text{patches}} \leftarrow \mathcal{G}$
2: $S \leftarrow \text{SamplePatchesFromVideo}(\mathcal{V}, \mathcal{G}, \mathcal{H})$
3: $\mathbf{D}_{\text{patches}} \leftarrow \mathbf{D}_{\text{patches}} \cup S$
4: $\mathbf{D} \leftarrow \mathcal{H}(\mathbf{D}_{\text{patches}})$
5: $\mathcal{F} = \text{LearnDeepFeatures}(\mathbf{D})$
6: $\mathcal{C} = \text{TrainSVM}(\mathcal{F}(\mathcal{H}(G_{\text{pos}})), \mathcal{F}(\mathcal{H}(G_{\text{neg}})))$
7: **return** $\{\mathcal{F}, \mathcal{C}\}$

Algorithm 2 Biased sampling of patches from video

Input: $\mathcal{V}$, $\mathcal{G}$, $\mathcal{H}$
Output: Sampled patches, $\mathbf{D}_{\text{patches}}$
1: $C = \text{TrainSVM}(\mathcal{H}(G_{\text{pos}}), \mathcal{H}(G_{\text{neg}}))$
2: Sample N frames from $\mathcal{V}$.
3: Run sliding-window detector with C on the N sampled frames
4: $\mathbf{D}_{\text{patches}} \leftarrow$ positive detections from detector
5: **return** $\mathbf{D}_{\text{patches}}$

Algorithm 3 Deep Feature Learning

Input: HOG features, $\mathbf{D}$
Output: Learnt non-linear projection function, $\mathcal{F}$
1: Apply PCA on $\mathbf{D}$ and keep 99% variance
2: $\mathbf{D} \leftarrow$ Project $\mathbf{D}$ onto principal component space
3: ndimsIn $\leftarrow$ number of dimensions of $\mathbf{D}$
4: Estimate intrinsic dimension of $\mathbf{D}$ using [8]
5: ndimsOut $\leftarrow$ Estimated intrinsic dimension
6: $\mathcal{A} \leftarrow \text{SetUpAutoEncoder}(\text{ndimsIn}, \text{ndimsOut})$
7: $\mathcal{A} \leftarrow$ initialise $\mathcal{A}$ using [4]
8: $\mathcal{A} \leftarrow \text{Min LossFunc}(\mathcal{A}, \mathbf{D})$ using mini-batch L-BFGS
9: $\mathcal{F} \leftarrow$ Remove the decoder part of $\mathcal{A}$
10: **return** $\mathcal{F}$

2.2 Sampling representative data from video

Although we are not concerned with any supervision labels during the feature learning stage, we would like to get a mixture of both pedestrian and non-pedestrian patches from the video. However, naive random sampling of pedestrian-sized patches from video in the space of multi-scale sliding windows would result in pedestrian patches being sampled only with extremely low probability. Therefore, we propose a biased sampling strategy as given in Algorithm 2.

2.3 Deep feature learning

In order to learn deep non-linear features, we use a deep autoencoder. We do not use any layer-wise stacked pre-training for initialization since recent research [10, 9] has shown that one of the main problems with deep networks is *pathological curvature* which looks like local optima to 1^{st}-order optimization techniques such as gradient descent, but not to 2^{nd}-order optimization methods.

Therefore, combined with a sensible weight initialization proposed in [4], we use the limited-memory version of Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm which is an approximated 2^{nd}-order method. This also has the added advantage that there is no need to set or tune the learning rate. We use mini-batch training for efficiency and robustness. The formal description is given in Algorithm 3 and explained below.

Data normalization & intrinsic dimensionality estimation. We first perform PCA and keep 99% of the total variance to condition the data for faster convergence during subsequent deep learning. After projecting the data using PCA coefficients, we estimate the intrinsic dimensionality of the data using [8].

Setting up the deep autoencoder architecture & initialization. The architecture is shown in Fig. 2. Note that the network structure is obtained automatically and systematically and we do not have to manually tune it. After the network has been set up, we randomly initialize it using the method in [4].

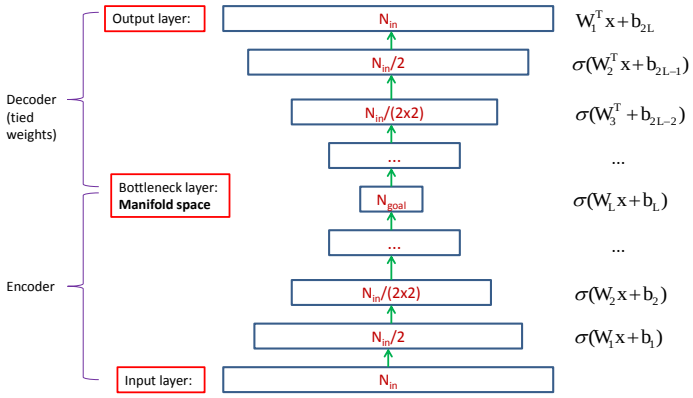

Fig. 2. The deep autoencoder architecture. N_{in} is the number of data dimensions after normalization. N_{goal} is the automatically estimated intrinsic dimensionality. Each hidden layer has half of the number of hidden neurons as its previous layer and this is repeated until N_{goal} is reached. After that, the decoding layers is mirrored to the encoding layers. The encoding and decoding parts are symmetric and weights are tied (but not the biases). All hidden layers have hyperbolic tangent nonlinearity activation (represented by $\sigma(\bullet)$). There are a total of L hidden nonlinear layers in the encoder (which produces a total of $2L$ layers feed-forward neural network.)

Network optimization. In order to train the autoencoder, the network in Fig. 2 can be mathematically written as a smooth differentiable multivariate loss function which should be minimized. This is given in Equation 1; m refers to the number of data in each mini-batch, W_j is affine projection matrix for layer j, $x^{(i)}$ is a column vector of data point i and b_j is a vector of bias for layer j.

$$\underset{\substack{W_1,\ldots,W_L,\\ b_1,\ldots,b_{2L}}}{\arg\min} \frac{1}{2m}\sum_{i=1}^{m}\Bigg\{\Bigg\|\sum_{j=1}^{L}\sigma(W_j x^{(i)}+b_j)+\sum_{j=L+1}^{2L-1}\sigma(W_{(2L-j+1)}^{T}x^{(i)}+b_j)+ W_1^T x^{(i)}+b_{(2L)}-x^{(i)}\Bigg\|_2^2\Bigg\} \tag{1}$$

Removing the decoding part. After training the deep network, the decoder part is no longer needed and is thus removed. We now have a deep non-linear projection function $\mathcal{F}$ with the number of projection layers given by L.

2.4 Training scene-specific detector

We use the learnt encoder, $\mathcal{F}$, to project the generic dataset $\mathcal{G}$ and train a SVM on these features. This is illustrated in Fig. 3.

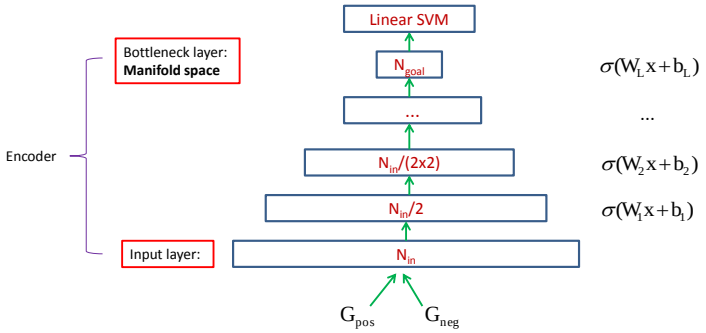

Fig. 3. Training the scene-specific detector.

3 Experimental Results

Datasets. INRIA pedestrian dataset [1] is used as the source dataset. We evaluate on two target datasets: CUHK Square (a 60 mins video) and MIT Traffic (90 mins) [12]. Frame samples are shown in Fig. 4. Each dataset is divided into two halves: the 1st half is used for unsupervised detector adaptation and the 2nd half for quantitative evaluation. These datasets are very challenging: they vary greatly from the INRIA dataset in terms of resolution, camera angle and poses.

Fig. 4. A frame sampled from the CUHK video (left) and from the MIT video (right).

Deep learning parameters. For CUHK experiments, the layer sizes for encoder part of the deep network are found to be $\mathcal{E} = [1498, 749, 375, 187, 94, 35]$. The decoder layer sizes are $\mathcal{D} = [94, 187, 375, 749, 1498]$. The complete layer sizes for the whole network is therefore given by $\mathcal{R} = [\mathcal{E}, \mathcal{D}]$. The first and last elements of $\mathcal{R}$ correspond to input and output layer sizes respectively (*i.e.* the number of dimensions after PCA projection) and the ones in the middle are hidden layers. In this case, 35 is the size of the bottleneck layer. For MIT, $\mathcal{E} = [1536, 768, 384, 192, 96, 48, 23]$. Mini-batch size for training is fixed at 1000 for both CUHK and MIT.

Evaluation and discussion. We use precision-recall (PR) curves in order to compare the performance. Detection bounding boxes are scored according to the PASCAL 50% overlap criteria [3]. Average Precision (AP) is calculated for each PR curve by integrating the area under the curve. For each target dataset, we perform 3 different types of experiments:

1. `Generic`: The detector (HOG+SVM) trained on INRIA dataset. This is the baseline *without* any domain adaptation.
2. `Geodesic(CVPR12)`: The approach proposed by Gong *et al.* [5]. We use the code made available by them and extend it to be applicable to video.
3. `Proposed`: Our proposed detector adaptation algorithm.

The PR curves are shown in Fig. 5 and the corresponding APs are shown in Table 1. As can be seen, our algorithm (`Proposed`) outperforms the baseline (`Generic`) and the state-of-art (`Geodesic`) in both datasets. For CUHK, `Proposed` achieves almost twice the AP of baseline (0.6880 vs. 0.3554), whereas `Geodesic` has less improvement over the baseline (0.5286 vs. 0.3554). Interestingly, for MIT, `Geodesic` performs worse than the baseline whereas `Proposed` clearly improves over the baseline and attains an AP which is around 1.5 times that of the baseline. This suggests that for the MIT traffic dataset, which is a more difficult dataset than the CUHK dataset (partly due to much lower resolution), the state-of-the-art `Geodesic` fails to improve over the baseline whereas

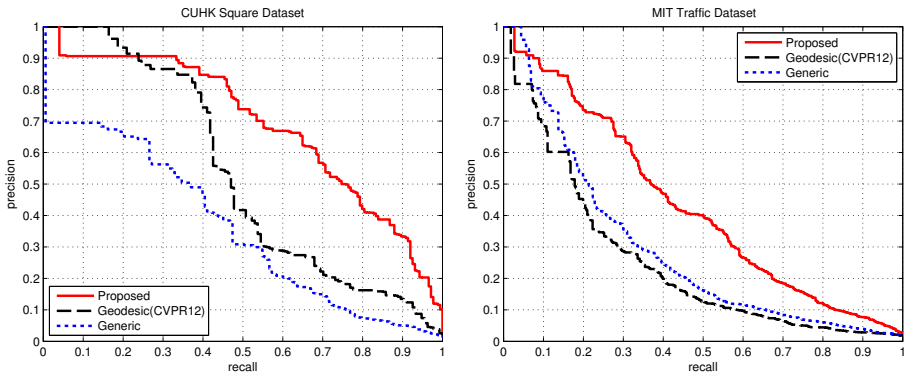

Fig. 5. Detection precision-recall curves

our approach, due to unsupervised learning of deep non-linear features and the resulting implicit manifold regularization, achieves much better results.

Table 1. Average precision results

	CUHK (AP)	MIT (AP)
`Generic`	0.3554	0.2883
`Geodesic(CVPR12)` [5]	0.5286	0.2460
`Proposed`	**0.6880**	**0.4292**

4 Conclusion

In this paper, we propose an algorithm to automatically generate a scene-specific pedestrian detector that is tuned to a particular scene by unsupervised domain adaptation of a generic detector. Our algorithm learns the underlying manifold where both the generic and the target dataset jointly reside and a detector is trained in this space, implicitly regularized to perform well on the target scene. Evaluation on two public video datasets show the effectiveness of our approach.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (1). pp. 886–893 (2005)
2. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: CVPR. pp. 304–311 (2009)
3. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88(2), 303–338 (2010)
4. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)
5. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: CVPR. pp. 2066–2073 (2012)
6. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: ICCV. pp. 999–1006 (2011)
7. Kevin Tang, Vignesh Ramanathan, L.F.F.D.K.: Shifting weights: Adapting object detectors from image to video. In: Neural Information Processing Systems (NIPS) (2012)
8. Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. In: Advances in neural information processing systems. pp. 777–784 (2004)
9. Martens, J.: Deep learning via hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 735–742 (2010)
10. Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A., Le, Q.V.: On optimization methods for deep learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 265–272 (2011)
11. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference pp. 1521–1528. IEEE (2011)
12. Wang, M., Li, W., Wang, X.: Transferring a generic pedestrian detector towards specific scenes. In: CVPR. pp. 3274–3281 (2012)