Domain Adaptation for Pedestrian Detection

by

Kyaw Kyaw Htike

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.

UNIVERSITY OF LEEDS

The University of Leeds
School of Computing

July 2014

Declarations

The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Kyaw Kyaw Htike and David Hogg, "Unsupervised detector adaptation by joint dataset feature learning", *International Conference on Computer Vision and Graphics*, ICCVG, 2014. Chapter 4 of the thesis is based on the contents of this paper.
Kyaw Kyaw Htike and David Hogg, "Efficient non-iterative domain adaptation of pedestrian detectors to video scenes", *International Conference on Pattern Recognition*, ICPR, 2014. Chapter 5 of the thesis is based on the contents of this paper.
Kyaw Kyaw Htike and David Hogg, "Weakly supervised pedestrian detector training by unsupervised prior learning and cue fusion in videos", *International Conference on Image Processing*, ICIP, 2014. Chapter 6 of the thesis is based on the contents of this paper.

The candidate confirms that the role of the second author in the jointly-authored publications listed above was purely supervisory.

Abstract

Object detection is an essential component of many computer vision systems. The increase in the amount of collected digital data and new applications of computer vision have generated a demand for object detectors for many different types of scenes digitally captured in diverse settings. The appearance of objects captured across these different scenarios can vary significantly, causing readily available state-of-the-art object detectors to perform poorly in many of the scenes. One solution is to annotate and collect labelled data for each new scene and train a *scene-specific* object detector that is specialised to perform well for that scene, but such a method is labour intensive and impractical.

In this thesis, we propose three novel contributions to learn scene-specific pedestrian detectors for scenes with minimal human supervision effort. In the first and second contributions, we formulate the problem as *unsupervised domain adaptation* in which a readily available *generic* pedestrian detector is automatically adapted to *specific* scenes (without any labelled data from the scenes). In the third contribution, we formulate it as a *weakly supervised learning* algorithm requiring annotations of only pedestrian centres.

The first contribution is a detector adaptation algorithm using *joint dataset feature learning*. We use state-of-the-art deep learning for the purpose of detector adaptation by exploiting the assumption that the data lies on a low dimensional manifold. The algorithm significantly outperforms a state-of-the-art approach that makes use of a similar manifold assumption.

The second contribution presents an efficient detector adaptation algorithm that makes effective use of cues (*e.g.* spatio-temporal constraints) available in video. We show that, for videos, such cues can dramatically help with the detector adaptation. We extensively compare our approach with state-of-the-art algorithms and show that our algorithm outperforms the competing approaches despite being simpler to implement and apply.

In the third contribution, we approach the task of reducing manual annotation effort by formulating the problem as a weakly supervised learning algorithm that requires annotation of only approximate centres of pedestrians (instead of the usual precise bounding boxes). Instead of assuming the availability of a generic detector and adapting it to new scenes as in the first two contributions, we collect manual annotation for new scenes but make the annotation task easier and faster. Our algorithm reduces the amount of manual annotation effort by approximately four times while maintaining a similar detection performance as the standard training methods. We evaluate each of the proposed algorithms on two challenging publicly available video datasets.

Acknowledgements

First and foremost, I would like thank my supervisor Prof. David Hogg for his guidance, advice and feedback throughout my PhD. I would also like to thank him and the department for funding me for various events such as a summer school and conferences during the PhD.

My sincere gratitude goes to Postgraduate Scholarships at University of Leeds for the Fully-Funded International Research Scholarship (FIRS) which funded me for three years of my PhD. Without FIRS, I would not have had the opportunity to come to UK and do my PhD at this university.

I thank colleagues and staff at School of Computing for a lot of fruitful discussions and support, and for providing an environment conducive to carrying out research.

Last but not least, I take this opportunity to thank my parents and family in Malaysia for their love, patience, encouragement and support.

Notations

The following table is a list of important math notations in the thesis.

Domain and task	
$\mathcal{D}$	A domain.
$\mathcal{D}_s$	A source domain.
$\mathcal{D}_t$	A target domain.
$\mathcal{X}$	A feature space.
$\mathcal{X}_s$	A source feature space.
$\mathcal{X}_t$	A target feature space.
$\mathcal{T}$	A task.
$\mathcal{T}_s$	A source task.
$\mathcal{T}_t$	A target task.
$\mathcal{Y}$	A label space.
$\mathcal{Y}_s$	A source label space.
$\mathcal{Y}_t$	A target label space.
Datasets	
$\mathbf{x}$	A feature (column) vector (denoting a single data point). $\mathbf{x} \in \mathbb{R}^k$ where k is the length of the feature vector.
y	Supervision (class) label associated with a training data point, *e.g.* $y \in \{\text{pedestrian}, \text{non-pedestrian}\}$.
$\mathbf{X}$	A training dataset which is a set of feature vectors $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$. The symbol $\mathbf{X}$ is for general usage and could denote a labelled or an unlabelled dataset depending on the context. If the dataset is labelled, then $\mathbf{X}$ is associated with a set of class labels.
$\mathbf{Y}$	A set of class labels corresponding to a labelled training dataset. $\mathbf{Y} = \{y_1, y_2, \dots\}$.
$\mathbf{X}_s^l$	Source labelled dataset. A set of feature vectors in the source domain and each of the feature vectors is associated with a class label which is given by the corresponding element of the set $\mathbf{Y}_s$.
$\mathbf{Y}_s$	A set of class labels corresponding to $\mathbf{X}_s^l$.
$\mathbf{X}_t^l$	Target labelled dataset. A set of feature vectors in the target domain and each of the feature vectors is associated with a class label which is given by the corresponding element of the set $\mathbf{Y}_t$.
$\mathbf{Y}_t$	A set of class labels corresponding to $\mathbf{X}_t^l$.

$\mathbf{X}_t^u$	Target unlabelled dataset, *i.e*. a set of feature vectors in the target domain. There is no label information associated with the feature vectors.
$\mathbf{X}_t^+$	A set of feature vectors corresponding to the positive class (*e.g*. pedestrian class) in the target domain.
$\mathbf{X}_t^-$	A set of feature vectors corresponding to the negative class (*e.g*. non-pedestrian class) in the target domain.
$\mathbf{X}_{s+t}^u$	Combination of the source and the target datasets (*i.e*. the union of the sets of feature vectors from the source and target domains). Labels from the source dataset are not considered. The resulting dataset $\mathbf{X}_{s+t}^u$ is unlabelled and therefore no label information is attached.
$\mathbf{X}_t^{\text{fg}}$	A set of feature vectors obtained from all (unknown) categories of foreground objects in the target scene (*i.e*. video).
$\mathbf{X}_t^{\text{bg}}$	A set of feature vectors obtained from background structures in the target scene (*i.e*. video).
Other Sets or Sequences	
$\mathbb{V}$	A video, *i.e*. a sequence of images $[I_1, I_2, \dots]$.
$\mathbb{P}$	A set of (colour) image patches $\{p_1, p_2, \dots\}$ where $p_i \in \mathbb{R}^{m \times n \times 3}$.
$\mathbb{B}$	A set of bounding boxes $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ where $\mathbf{b}_i$ holds both the bounding box information (*e.g*. top-left corner, width and height of the bounding box) and the frame number in the video where the bounding box is located.
$\mathbb{W}$	A set of multi-scale sliding windows within an image $\{\mathbf{w}_1, \mathbf{w}_2, \dots\}$.
$\mathbb{S}_{\text{weak}}$	Set of weak supervisions in the form of approximate centre locations of objects $\{\mathbf{cc}_1, \mathbf{cc}_2, \dots\}$.
$\mathbb{S}_{\text{bbox}}$	Set of generated bounding box annotations $\{\mathbf{bb}_1, \mathbf{bb}_2, \dots\}$ obtained from the set of weak supervisions.
Functions	
$\mathcal{C}_s$	Generic (*i.e*. source) classifier/detector; $\mathcal{C}_s : \mathbb{R}^k \rightarrow \mathbb{R}$, where k is the length of the input feature vector. The function takes as input a feature vector and outputs a classification score. The classification score can be thresholded to give a binary classification decision, *i.e*. the function can also be written as $\mathcal{C}_s : \mathbb{R}^k \rightarrow \{1, 0\}$, where "1" refers to the positive class and "0" refers to the negative class.

$\mathcal{C}_t$	Scene-specific (*i.e.* target) classifier/detector; $\mathcal{C}_t : \mathbb{R}^k \rightarrow \mathbb{R}$, where k is the length of the input feature vector. The function takes as input a feature vector and outputs a classification score. The classification score can be thresholded to give a binary classification decision, *i.e.* the function can also be written as $\mathcal{C}_t : \mathbb{R}^k \rightarrow \{1,0\}$, where "1" refers to the positive class and "0" refers to the negative class.
$\mathcal{C}_p$	An unsupervised object prior for a video/scene in the form of a classifier; $\mathcal{C}_p : \mathbb{R}^k \rightarrow [0,1]$, where k is the length of the input feature vector. It gives the (prior) probability (within the range 0 and 1 inclusive) that a given data point (in the form of a feature vector) is an object (of any class apart from the background) in the scene.
$\mathcal{H}$	Function for feature extraction given raw pixel values to get "base features"; $\mathcal{H} : \mathbb{R}^{m \times n \times 3} \rightarrow \mathbb{R}^k$, where m and n are the number of rows and columns of the input colour image patch respectively and k is the length of the feature vector. For example, $\mathcal{H}$: a color image patch $\rightarrow$ HOG feature vector.
$\mathcal{A}$	A deep autoencoder architecture represented as a function; $\mathcal{A} : \mathbb{R}^k \rightarrow \mathbb{R}^k$, where k is the length of the input feature vector. It first projects the input data through nested non-linear functions and then decodes the encoded data through nested non-linear functions in an attempt to reconstruct the original data.
$\mathcal{F}$	Function to project a feature vector into manifold space (obtained by deep learning); $\mathcal{F} : \mathbb{R}^k \rightarrow \mathbb{R}^{\hat{k}}$, where k is the length of the input feature vector and $\hat{k}$ is the dimension of the manifold (*i.e.* the *intrinsic dimension*) and $\hat{k} \ll k$.
σ	Activation function of a neural network; $\sigma : \mathbb{R} \rightarrow \mathbb{R}$.
$\mathcal{N}$	Function that gives the likelihood of a bounding box window to be consistent with the weak supervision; $\mathcal{N} : \mathbb{R}^4 \rightarrow \mathbb{R}$.

Apart from the notations listed above, in general, the following rules are used:

- Bolded capital letters denote matrices. For instance, $\mathbf{W}$ represents a weight matrix for affine projection.
- Bolded small letters (*e.g.* $\mathbf{m}$) denote column vectors. A row vector is denoted by its transpose, *e.g.* $\mathbf{m}^T$.
- Non-bolded small letters are for scalars, *e.g.* i, j and k.
- Capital whiteboard-style letters (such as $\mathbb{V}$ and $\mathbb{S}$) are for sets or sequences.

Contents

1 Introduction **1**

1.1 Challenges . . . 1
1.2 Research Questions . . . 5
1.3 Thesis Contributions . . . 6
1.4 Thesis Outline . . . 7

2 Background **9**

2.1 Introduction . . . 9
2.2 Pedestrian Detection . . . 9
2.2.1 Building a Pedestrian Model . . . 10
2.2.2 Detecting Pedestrians with the Learnt Model . . . 12
2.2.2.1 Hypothesis generation and scoring . . . 13
2.2.2.2 Combining the scored hypotheses . . . 14
2.3 Feature Extraction . . . 14
2.4 Regularized Linear Classifiers . . . 16
2.5 Transfer Learning . . . 18
2.6 Domain Adaptation . . . 21
2.6.1 Domain Adaptation vs. Semi-supervised Learning . . . 22
2.7 Weakly Supervised Learning . . . 24

3 Literature review **26**

3.1 Introduction . . . 26
3.2 Domain Adaptation . . . 28
3.2.1 Domain Adaptation for Image Classification . . . 29
3.2.2 Domain Adaptation for Object Detection in Videos . . . 32
3.2.3 Areas Related to Domain Adaptation . . . 38
3.2.3.1 Learning moving object detectors . . . 38
3.2.3.2 Semi-supervised learning for object detection . . . 41

3.3 Weakly Supervised Learning For Object Detection . . . 43
3.4 Relation of this Thesis to Prior Research . . . 45
3.4.1 First Thesis Contribution (Chapter 4) . . . 45
3.4.2 Second Thesis Contribution (Chapter 5) . . . 46
3.4.3 Third Thesis Contribution (Chapter 6) . . . 48

4 Unsupervised Detector Adaptation by Joint Dataset Feature Learning 50
4.1 Overview . . . 50
4.2 Contributions . . . 51
4.3 Proposed Approach . . . 51
4.3.1 Overview . . . 51
4.3.2 Sampling Representative Unlabelled Data from Target Video . . . 53
4.3.3 Learning the Manifold of the Joint Unlabelled Dataset . . . 54
4.3.3.1 Data normalisation . . . 57
4.3.3.2 Setting up the deep autoencoder architecture . . . 58
4.3.3.3 Initialising the deep autoencoder architecture . . . 60
4.3.3.4 Network optimization . . . 61
4.3.3.5 Removing the decoding part . . . 62
4.3.4 Training the Scene-specific Detector . . . 63
4.4 Experimental Results . . . 63
4.4.1 Datasets . . . 63
4.4.2 Base Features . . . 65
4.4.3 Unlabelled Patches Sampled for Deep Learning . . . 65
4.4.4 Deep Learning Parameters . . . 65
4.4.5 Implementation Details . . . 65
4.4.6 Evaluation & Discussion . . . 66
4.5 Conclusion . . . 73

5 Efficient Non-iterative Domain Adaptation of Pedestrian Detectors 74
5.1 Overview . . . 74
5.2 Introduction . . . 74
5.3 Contributions . . . 76
5.4 Our Approach . . . 77
5.4.1 Overview . . . 77
5.4.2 Generating Bounding Box Proposals . . . 77
5.4.3 Initial Verification . . . 77
5.4.4 Spatio-temporal Verification & Expansion . . . 79

5.4.4.1 Details on short-term tracking . . . 82
5.4.5 Collecting Negative Examples . . . 82
5.4.6 Training the Scene-specific Detector . . . 82
5.4.7 Analysis on Bounding Box Proposal & Verification . . . 83
5.4.8 Analysis on Spatio-temporal Verification & Expansion . . . 84
5.5 Experimental Results . . . 85
5.5.1 Classifier & Features . . . 85
5.5.2 Datasets . . . 85
5.5.3 Descriptions of Experiments . . . 85
5.5.4 Evaluation . . . 88
5.5.4.1 Comparison with generic detector . . . 88
5.5.4.2 Comparison with state-of-the-art . . . 88
5.5.4.3 Comparison with human supervision . . . 89
5.5.4.4 Effect of throwing away the source dataset . . . 89
5.5.4.5 Effect of training with another type of classifier . . . 89
5.5.5 Analysis of Failure Modes . . . 89
5.6 Conclusion . . . 94

6 Weakly Supervised Detector Training by Unsupervised Prior Learning and Cue Fusion 95
6.1 Overview . . . 95
6.2 Contributions . . . 97
6.3 Our Approach . . . 97
6.3.1 Unsupervised Pedestrian Prior Learning . . . 99
6.3.2 Cue Fusion and Optimization . . . 102
6.3.3 Training the Scene-specific Detector . . . 105
6.4 Experimental Results . . . 109
6.5 Conclusion . . . 113

7 Conclusion and Future Work 114
7.1 Overall Discussion . . . 114
7.2 Summary of Strengths and Weaknesses, and Recommendations . . . 116
7.3 Conclusion . . . 118
7.4 Summarised Answers to Research Questions . . . 118
7.5 Limitations . . . 119
7.6 Future Work . . . 120
7.7 Publications . . . 121

Bibliography **122**

List of Figures

1.1 Random samples from some generic pedestrian datasets (only pedestrians, *i.e*. positive examples, are shown). 2
1.2 Random samples from scene-specific pedestrian datasets (only pedestrians, *i.e*. positive examples, are shown). 4

2.1 An example of pedestrian detection in an image. 10
2.2 A general system for training a pedestrian detector. 11
2.3 A general system for training a pedestrian detector (with feature extraction explicitly shown). 12
2.4 Visualization of HOG features. 15
2.5 Transfer learning between two tasks. 18
2.6 Some samples of cats and dogs from the PASCAL VOC dataset. 20
2.7 Samples of faces from source and target domains. 23
2.8 An example of the comparison between standard supervision and weak supervision. 25

3.1 The need for domain adaptation for image classification. 29
3.2 Multi-domain object database to study and evaluate domain adaptation algorithms for image classification. 30
3.3 Illustration of sampling points between the subspaces of the source and target domains on the Grassmann manifold. 31
3.4 Geodesic flow kernel to model the gradual change from the source domain to the target domain. 31
3.5 Detector adaptation approach of Kemhavi *et al*. [58]. 33
3.6 System overview of the algorithm of Wang *et al*. [113]. 34
3.7 An iterative self-training technique of Wang and Wang [111]. 37
3.8 Office corridor scene used in [75]. 39
3.9 Indoor scene for pedestrian detection in [46]. 39
3.10 Weakly supervised learning approach of Galleguillos *et al*. [38]. 42

3.11 Weakly supervised learning approach of Weber *et al*. [114]. 43
3.12 Overview of the weakly supervised learning approach by Prest *et al*. [84]. 44

4.1 Overview of the proposed algorithm for joint dataset feature learning. . . 52
4.2 The deep autoencoder architecture. 58
4.3 Hyperbolic tangent activation function to impart nonlinearity in the deep network. 60
4.4 Training the scene-specific detector. The deep autoencoder has the decoder part removed. 62
4.5 Frame samples from the CUHK video (left) and from the MIT video (right). 64
4.6 Effect of changing the number of samples from target dataset. 66
4.7 The effectiveness of the proposed biased sampling. 69
4.8 Objective error function value decreasing over iterations. 70
4.9 Detection recall-FPPI curves 71
4.10 Effect of training with a Random Forest in the manifold space. 72

5.1 The high level view of Chapter 5. 75
5.2 Overview of our proposed efficient detector adaptation approach 79
5.3 Random samples of patches from bounding box proposals and verified proposals. 80
5.4 Visualization of two example tracklets obtained from tracking two verified proposals respectively. 81
5.5 Detection performance curves (all on testing datasets). 87
5.6 Effect of training with a Random Forest. 90
5.7 Samples of false positives and false negatives in CUHK dataset. 92
5.8 Samples of false positives and false negatives in MIT dataset. 93

6.1 Strong versus weak annotation. 96
6.2 Overview of the proposed weakly supervised learning algorithm 98
6.3 CUHK Square dataset: Random samples of patches corresponding to foreground objects. 106
6.4 MIT Traffic dataset: Random samples of patches corresponding to foreground objects. 107
6.5 Bivariate normal distribution. 108
6.6 Visualisation of 100 samples from the bivariate normal distribution. . . . 108
6.7 Detection performance curves for CUHK (left) and MIT (right) 111
6.8 Cost comparison 112

7.1 Comparison of the main novel contributions in the thesis. 115

List of Tables

2.1 Annotation summary for transfer learning 20

4.1 Video dataset details 63
4.2 Deep learning architecture parameters 66

7.1 Summary of the strengths and weaknesses of the proposed methods. . . . 117

Chapter 1

Introduction

Computer vision is about automatic analysis and understanding of visual data (such as images and videos) to extract useful information. There are many sub-areas within the field of computer vision, one of which is object detection which forms the foundation of many intelligent scene understanding systems. Due to its importance, object detection has received a lot of attention in computer vision [3]. Pedestrian detection in particular plays an important role in real world outdoor scenes especially in urban areas. In this thesis, although the proposed algorithms could potentially be used for learning detectors for any object category (such as pedestrians, cars, buses and bicycles), we focus on the task of pedestrian detection since pedestrians are of most interest in many applications of computer vision.

1.1 Challenges

Pedestrian detection in monocular images is a challenging task and a lot of progress has been made in this area (see [27,29,41] for reviews and comparisons). Most state-of-the-art pedestrian detectors require a supervised training stage based on a labelled dataset that is obtained from manual annotation of pedestrians (*e.g.* delineation of pedestrians by bounding boxes) and a sufficient number of non-pedestrian images [21, 27, 42]. The objective of the labelled dataset is to provide the classifier (being learnt) with large intra-class variations of pedestrians and non-pedestrians so that the resulting classifier is *generalisable*

(a) INRIA dataset [21] (b) Daimler dataset [29] (c) Caltech dataset [26]

Figure 1.1: Random samples from some generic pedestrian datasets (only pedestrians, *i.e.* positive examples, are shown).

to never-before-seen test data. This *generalisation* property is a sought-after property for most machine learning classification and regression tasks. When training a pedestrian detector, the goal is often: "Given *any* unseen test image, the detector should locate all the pedestrians in the image". In this thesis, we term such a detector as a *generic (pedestrian) detector* and the training data from which the detector was trained as a *generic (pedestrian) dataset*.

For a generic dataset, collected positive and negative examples are not (deliberately) limited to a particular scene and viewpoint and the aim of such a dataset is to collect as many variations of pedestrians as possible to produce detectors which should ideally perform well for any unseen test data. Examples of generic pedestrian datasets are INRIA Person Dataset [21], Daimler Mono Pedestrian Detection Benchmark Dataset [29] and Caltech Pedestrian Dataset [26]. The INRIA dataset consists of images of upright people taken from a variety of personal image collections. Pedestrian training data of the Daimler and the Caltech datasets are extracted from videos recorded with on-board cameras in vehicles being driven around various places in urban traffic. All these datasets consist of training data from a variety of scenes and places and as a result, the intra-class variations of pedestrians in such datasets is large. Figure 1.1 illustrates this observation.

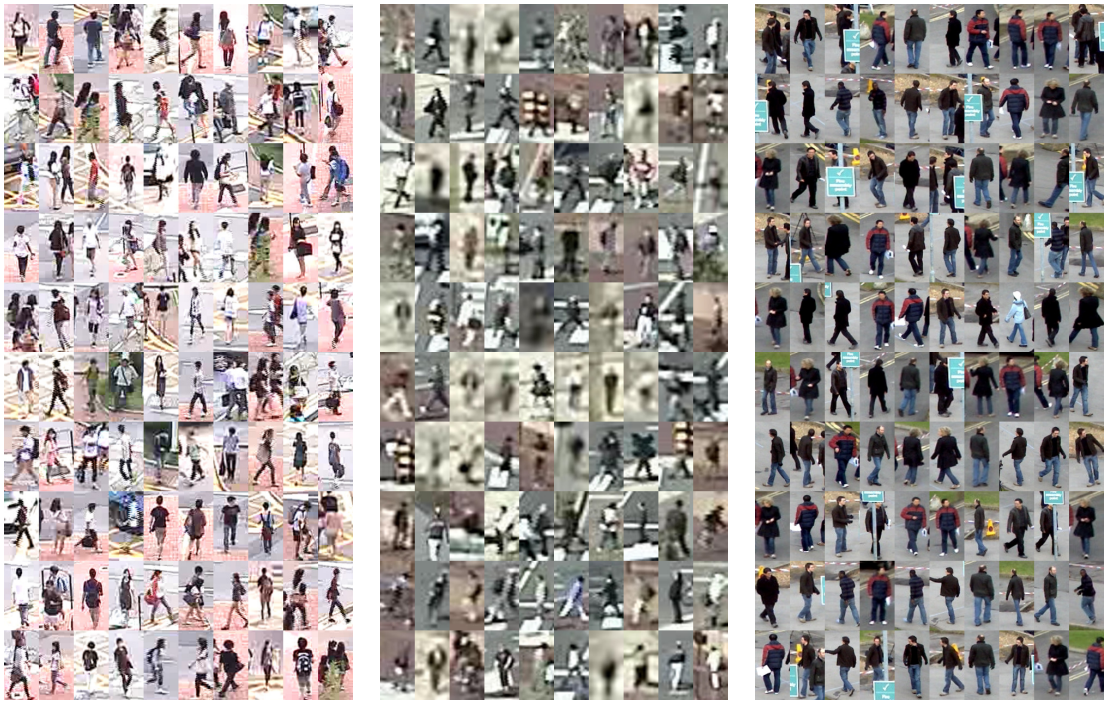
Despite the large intra-class variations present in such generic datasets, each of these datasets still has its own inherent bias. For example, since the INRIA dataset is taken from mostly personal digital image collections, many of the people in the training dataset are likely to be intentionally posing for cameras. This may be different from natural pedestrian poses and activities in real-life situations. For the Daimler and Caltech datasets, the pedestrians in the training set are biased to view-points and angles that cameras on-board vehicles could capture. Moreover, pedestrians from these datasets are taken from static images that have been captured using cameras fixed near the same ground plane as the captured pedestrians. This may be considerably different from situations where images of pedestrians are captured by video cameras looking down on a scene (e.g. surveillance videos).

This dataset bias has been recently studied by Torralba and Efros [105]. No dataset can possibly cover a *representative* set of all the possible variations of pedestrians and non-pedestrians the detector is likely to face at test time. As shown by [26, 91], detectors may fail to perform satisfactorily when applied to scenes that differ from the original training data in many aspects such as:

- Pedestrian pose
- Image or video resolution
- View-point
- Lighting condition
- Image or video compression effects
- Presence of motion blur

Furthermore, apart from this dataset problem, even assuming that there is a perfect generic dataset, it is non-trivial to learn a classifier that is "good" enough to capture all these highly complex and multi-modal variations of the dataset whilst at the same time not over-fit on the training data. In addition, for most of the pedestrian detectors, the speed of the detector is an important criterion that has to be taken into consideration, particulary since a classifier must be applied on many (typically hundreds of thousands of) multi-scale sliding windows in each image. This rules out time-consuming feature extraction mechanisms and complex classifiers.

It is, however, crucial to ask the question of whether, deployed pedestrian detectors in real-life actually need to work well across any test data. The short answer is that for



(a) CUHK Square dataset [110] (b) MIT Traffic dataset [111] (c) PETS 2009 dataset [35]

Figure 1.2: Random samples from scene-specific pedestrian datasets (only pedestrians, *i.e*. positive examples, are shown).

most situations, they do not. Each deployed pedestrian detector needs to work well only for the specific scene and conditions that it is applied to. Given a particular scene, the intra-class variation of the pedestrians being captured by a fixed camera is limited compared to general situations. For example, due to the fixed camera angle, the view-point is fixed and the space of possible poses that a pedestrian can exhibit is a small subset of all the possible pedestrian poses. Furthermore, the lighting variation is also smaller and the image compression effects are similar for all the pedestrians captured by the same camera. In addition, the environment, the background and the surroundings are fixed which translate to less variation in non-pedestrian classes of data. Moreover, there are geographical, cultural and social constraints under a fixed location which, for example, may make pedestrians more likely to conform to similar styles of clothing. Further, most state-of-the-art detectors work by extracting features from a rectangular window and applying the learnt classifier. This means that pixels that do not correspond to pedestrians (also known as "scene context") are also inside the window. For a particular scene, this scene context can be captured effectively. Overall, the intra-class variation of pedestrians in a specific scene is smaller than the intra-class variation of pedestrians across all possible scenarios as shown in Figure 1.2.

Therefore, it seems that the solution then is to collect labelled data for each new scene specifically tailored for that scene. The resulting detector can be termed as a *scene-specific detector* since the detector is tuned and specialised to work well in a particular type of scene. The task is now clearly simpler: given any unseen test image in *this* scene, the detector should locate all the pedestrians in the image. This is a simpler task than building a generic detector.

There are two observations that can be made. Firstly, with the feature extraction mechanism and the classifier type held fixed, a scene-specific detector can be more accurate than a generic detector. Secondly, with the detector accuracy held fixed, the feature extraction mechanism and the classifier of the scene-specific detector can be simpler and faster than a generic detector due to having to learn to perform classification for a simpler task. This is clearly critical in real-time or embedded-processor applications. In this thesis, we make use of the first observation.

Although training a scene-specific detector that is specialised to each new scene seems like a good idea, in practice, it can be labour-intensive especially when considering the number of different scenes and applications for which we need pedestrian detectors. In this thesis, we address the problem by reducing the human supervision effort involved in learning scene-specific pedestrian detectors.

We achieve this in two ways. Firstly, we utilise the fact that although generic pedestrian detectors may not work well for specific scenes, they contain information about pedestrians in general and can be used as prior knowledge in the process to learn scene-specific detectors. We therefore formulate the task of learning scene-specific detectors as an *unsupervised domain adaptation* problem and contribute two novel algorithms that start 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. Secondly, we tackle the situation where a generic dataset is not available. For this, we contribute a novel algorithm to reduce human supervision effort by formulating the problem in a weakly supervised learning framework.

1.2 Research Questions

In this thesis, the following research questions are addressed.

1. What are the effective methods to adapt a generic pedestrian detector to specific scenes?

2. Can we perform the domain adaptation for pedestrian detection by using only the

manifold[1] assumption of data? Is there any benefit in using state-of-the-art deep learning to learn the manifold for the detector adaptation task for videos? How does this compare to other state-of-the-art detector adaptation algorithms using a similar manifold assumption?

3. For videos where spatio-temporal continuity information and other types of knowledge can be exploited, can we forego the explicit manifold learning step and have an efficient algorithm that makes use of these spatio-temporal cues and gives higher adaptation performance?

4. Is there any other way of providing supervision that is easier and faster than bounding box annotation for videos?

1.3 Thesis Contributions

To answer the aforementioned research questions, we make the following novel contributions:

1. We present a detector adaptation algorithm using *joint-dataset feature learning* that exploits the manifold assumption of data. The algorithm does not require background subtraction or tracking and is applicable to either static images or videos. For feature learning, state-of-the-art deep learning is utilised. In addition, due to large class imbalance associated with random sampling of patches for unsupervised feature learning, a simple biased sampling approach is proposed to minimise the problem. An effective technique to automatically set the structure of the deep network is also discussed. It is experimentally shown that the proposed contribution outperforms a state-of-the-art technique that makes a similar manifold assumption about data.

2. For videos, unlike in static images, many cues and knowledge can be automatically learned due to the spatio-temporal consistency of video data. In particular, this time, we exploit background subtraction and tracking to help with the detector adaptation and propose a simple and effective non-iterative self-training algorithm. Firstly, we introduce the idea of *bounding box proposals* and *initial verification* for efficient generation of a large number of scene-specific pedestrian data with a high probability of accuracy. Secondly, we propose the concept of *spatio-temporal*

[1] A *manifold* "is a topological space that is locally Euclidean" [92]. For example, circles and lines are one-dimensional manifolds. See Section 4.3.3 for details.

verification and expansion using short-term tracking. The proposed integrated system of the aforementioned concepts outperforms several state-of-the-art detector adaptation algorithms that adapt a generic pedestrian detector to videos captured in specific scenes. Unlike most state-of-the-art algorithms, the proposed algorithm does not require the presence of the generic data itself during detector adaptation; just the generic detector alone is sufficient.

3. A weakly supervised learning algorithm is proposed for reducing human supervision effort for pedestrian detection in videos. The algorithm only requires weak supervision in the form of approximate centre locations of pedestrians. The first sub-contribution is learning a pedestrian prior in an unsupervised way for a given video. The second sub-contribution is fusing the cues from the pedestrian prior and the provided weak supervision in an optimization framework. The proposed algorithm is efficient and can work with low resolution videos in which object part modelling and discovery are not feasible.

1.4 Thesis Outline

In Chapter 2, background material is given that explains the important concepts in the thesis, including pedestrian detection, feature extraction and classifiers and most importantly, an introduction to transfer learning, domain adaptation and weakly supervised learning. Domain adaptation is also briefly compared and contrasted with the more well known semi-supervised learning.

In Chapter 3, related works are discussed. We do not review literature on general object or pedestrian detection. Only studies that are close to the research in this thesis are included. The main attention is given to two areas: domain adaptation and weakly supervised learning as applied in the field of computer vision. Related works are discussed in detail, pointing out the strengths and the weaknesses of each approach where relevant. Moreover, the relation between the most relevant works and the thesis contributions (*i.e.* main algorithms proposed in this thesis) are also explained.

In each of Chapters 4, 5 and 6, we present a major contribution of the thesis, for a total of three major contributions. Chapters 4 and 5 represent the domain adaptation part of the thesis and Chapter 6 is about weakly supervised learning. In each of these chapters, we also list the contributions for the chapter. In addition, in each of these chapters, we present the methodology, experimental results, discussion on the results and a conclusion specific to the chapter.

In Chapter 7, the results corresponding to the proposed algorithms in Chapters 4, 5 and 6 are brought together and discussed. We then conclude the thesis by summarising the thesis and discussing the limitations and the future work that builds on the thesis. Furthermore, we also outline the answers to the research questions that are raised in Section 1.2.

Chapter 2

Background

2.1 Introduction

In this chapter, we describe the key background materials and concepts that are extensively used in and helpful to understand the rest of the thesis. Since we apply our domain adaptation and weakly supervised learning algorithms on the task of pedestrian detection, we first start with the explanation of what "pedestrian detection" is. Since the most common pipeline for pedestrian detection includes feature extraction and classification, we briefly introduce the concept of feature extraction and regularized linear classifiers. After this, we describe the main machine learning paradigms relevant to the thesis, namely transfer learning, domain adaptation and weakly supervised learning.

2.2 Pedestrian Detection

Pedestrian detection can be defined as "locating" pedestrians (if there are any) in images. The "location" goal is dependent upon the nature of the application and in this thesis, "location" refers to obtaining the position and spatial extent of pedestrians in images in the form of horizontally and vertically axis-aligned rectangles tightly fitting the pedestrians as shown in Figure 2.1. These are commonly known as *bounding boxes*. In some other applications (not addressed in this thesis), the "location" task could also refer to "pixel-wise localisation", *i.e*. getting a segmentation *mask* of pedestrians in images.

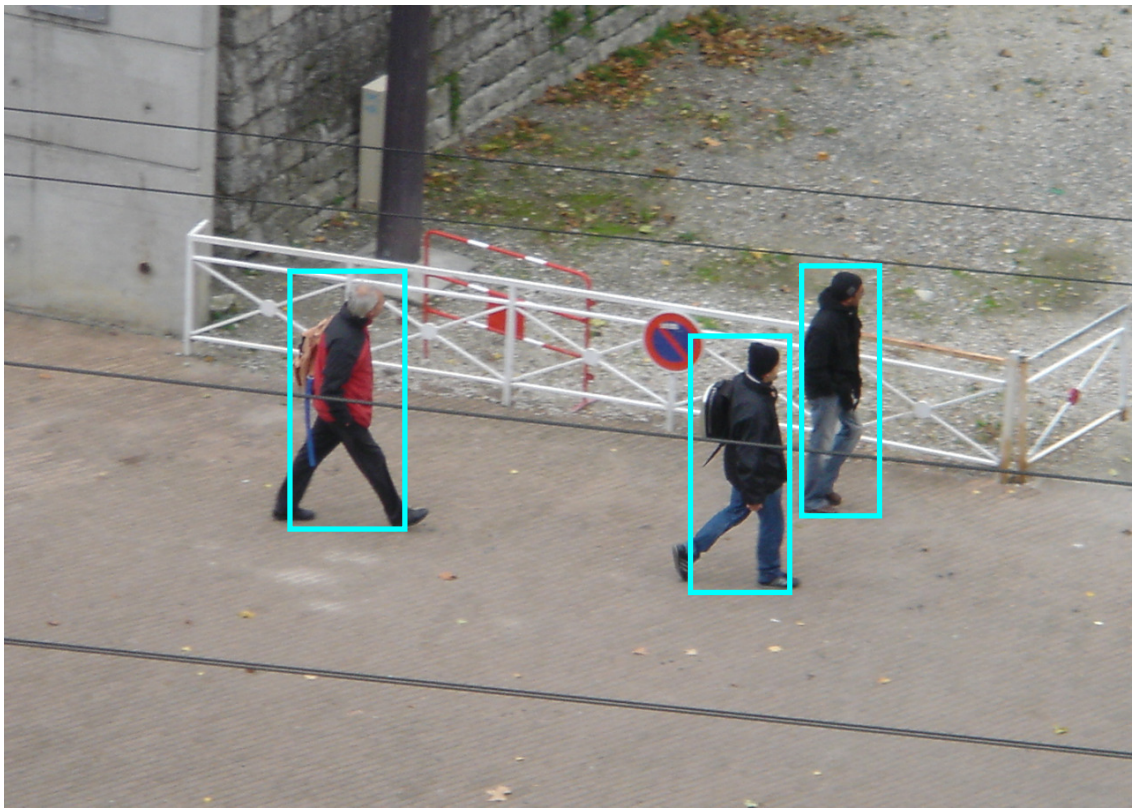


Figure 2.1: An example of pedestrian detection in an image.

There are 2 stages to a typical pedestrian detection task:

1. Building (*i.e.* training) a pedestrian model with a training dataset.
2. Applying the pedestrian model to detect pedestrians at test time.

2.2.1 Building a Pedestrian Model

For most successful detection systems, the first stage uses a machine learning approach in which some *examples* of pedestrians and non-pedestrians (which are known as *training examples* or *training dataset*) are manually collected and presented to a *learning* system that attempts to build a pedestrian *model* that generalises over the training examples. This generalisation property is often the most important goal of most machine learning systems since it allows the resulting model to correctly *predict* for unseen test data.

The training dataset, the learning system and the interplay between them are essential for obtaining a successful pedestrian model. Figure 2.2 shows the overview of a general system for training a pedestrian detector.

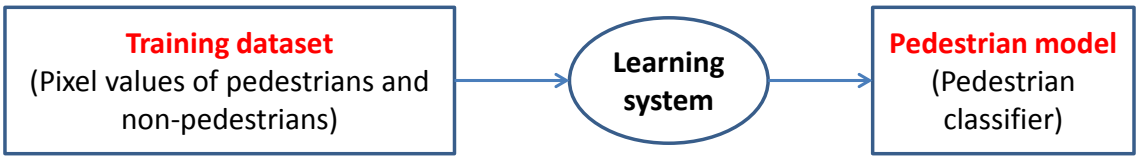


Figure 2.2: A general system for training a pedestrian detector.

For most machine learning tasks in real life, it is often too complex to learn (*i.e.* generalise) with the limited amount of labelled raw data to produce "good" models. Therefore, most machine learning systems make use of an additional step (before learning) commonly known as *feature extraction*.

The goal of feature extraction is to convert raw data into more "useful" features that are *invariant* to a certain set of transformations of the raw data. To be more precise, feature extraction can be seen as a step to *untangle* the raw data to produce features that are less dependent on each other and that are more robust to noisy changes in the raw data.

For example, if we are using raw image pixel values as input to a learning system for pedestrian detection, it is extremely difficult (if not outright impossible) to get a reasonable pedestrian model because raw pixel values are highly dependent on one another and raw pixel values of different pedestrians are only related to the *pedestrian class* (or the non-pedestrian class) in a very highly non-linear fashion.

Feature extraction is an attempt towards minimising this problem and from one perspective, it can be seen as a way of incorporating expert (human) knowledge into the learning process and from another perspective, it is putting a *prior* on the space of models.

Feature extraction can cover all the steps in the entire pipeline of converting the raw pixel values into a set of features. This can include various image pre-processing operations (such as illumination normalisation), edge detection, calculating image statistics of different orders and encoding relationships between different parts of the image.

In fact, the feature extraction step is arguably the most crucial step in building a useful machine learning system and it is considered as a research area in its own right in different computer vision tasks such as face detection, face recognition, pedestrian detection and activity classification. The task of building a pedestrian model can now be modified to explicitly include feature extraction as shown in Figure 2.3.

After feature extraction, a pedestrian model is *learnt* by *optimising* the parameters of a suitable objective function with respect to the given training dataset. The resulting pedestrian model is determined by the combination of the following factors:

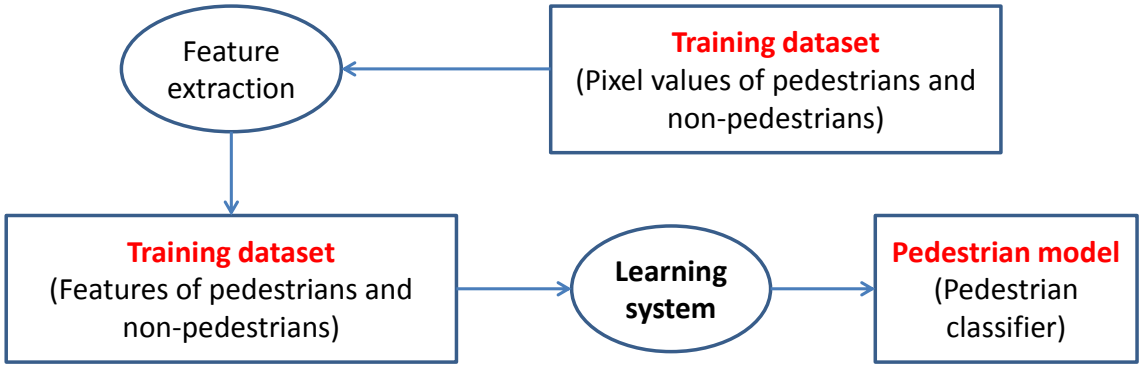


Figure 2.3: A general system for training a pedestrian detector (with feature extraction explicitly shown).

1. The quantity and quality of the raw training data.
2. The feature extraction algorithm.
3. The objective function for the optimisation (which defines the user-given family of models).
4. The optimisation algorithm (which chooses the "best" model out of the given family of models).

2.2.2 Detecting Pedestrians with the Learnt Model

After a pedestrian model has been learnt, pedestrian detection is the process of applying the model on a (novel) test image to locate pedestrians (in the form of bounding boxes around pedestrians). The exact process of how the model is applied is dependent upon the corresponding training algorithm. Generally, however, the task of detection can be divided into the following steps:

1. Hypothesis generation.
2. Testing (*i.e.* scoring) each hypothesis using the learnt pedestrian model (after any necessary feature extraction).
3. Combining the scored hypotheses to get the locations of the pedestrians.

These three steps may be purely *feedforward* in that the earlier step is not affected by the later step. In contrast, they can also be interdependent in the sense that hypothesis generation depends on the scores of (other) hypotheses and vice-versa. This can be used to speed up the process of hypothesis generation and scoring by pruning away some

hypotheses given knowledge of some other hypotheses. This is especially useful in part-based object detection (such as Deformable Part Models [33]) where the hypothesis space is very large. In that case, dynamic programming can be used to efficiently search for joint optimal placements of parts which fulfill both the part likelihoods and the likelihood of inter-part location relationships. However, non-part-based (*i.e.* monolithic) detectors have also be shown to benefit from exploiting this interdependency between hypothesis generation and scoring [14,24,47,48]. In the following subsections (and in this thesis), we shall focus on the former approach (*i.e.* feedforward) and on monolithic detectors where pedestrians are not (explicitly) modelled as connected parts.

2.2.2.1 Hypothesis generation and scoring

There are many approaches to hypothesis generation and scoring. The simplest way is the "exhaustive" approach. Despite its simplicity and being straightforward to implement, this is a very effective approach and is in fact the dominant paradigm in state-of-the-art object detection, commonly known as *sliding window detection*, where a fixed-sized rectangular window is scanned across the test image at finely sampled discrete locations on a grid. To detect objects of varying sizes, this process is repeated on several rescaled versions of the test image. At each window position, features from the patch corresponding to the window are extracted. This set of features serves as the input to the pedestrian model and the model outputs a score for that window position indicating the confidence of the model that the current patch being considered is a pedestrian.

Rather than an exhaustive search, hypothesis generation could also come from bottom-up image segmentation cues [1,15,28,69,87,106]. In order to make sure that the hypotheses generated in this way cover most of the pedestrians in an image, it is often necessary to run multiple segmentation algorithms and hierarchical merging of segments and collect the outputs of different segmentation algorithms at various levels of hierarchical merging as hypotheses [106]. This whole process may also be repeated in different colour spaces (such as RGB, greyscale, HSV and LAB) to achieve stability and robustness to changes in illumination, with the aim of increasing the probability of hypotheses containing actual objects. The number of hypotheses generated in this manner is expected to be an order of magnitude less than sliding window search. This type of hypothesis generation has the advantage that more complex and computationally expensive feature extraction and classification algorithms can be used due to having fewer hypotheses to score. However, the disadvantage is that hypothesis generation is dependent on the quality of the segmentation(s) and therefore might not work well in many types of situation (such as for low resolution images where segmentation is generally unreliable).

Sliding window hypothesis generation can also be limited to be within certain regions of interest (ROIs) or scales within an image [29]. These ROIs can be obtained in many ways. Firstly, they can be obtained by considering the prior information about a class of objects with respect to a scene. For instance, the fact that pedestrians tend to occupy only certain areas of the scene may be used as prior information and only these areas could be treated as ROIs. Moreover, the range of widths and heights of pedestrians can be specified a priori to decrease the number of hypotheses. Secondly, scene geometry and ground plane(s) can be automatically estimated which can then be used to constrain the hypothesis generation (*e.g.* [64]). Finally, motion estimation by optical flow or stereo vision [39, 117] can be used to highlight regions likely to contain moving objects and those regions can be treated as ROIs. Although the aforementioned techniques could help speed up object detection and potentially reduce false positives, they are sometimes not fully automatic (*e.g.* in having to manually specify prior information for different scenes) and they may require additional equipment and setup (such as stereo cameras) and expensive algorithms (such as online 3D estimation). Moreover, these approaches are complementary to traditional sliding window approaches (without using these ROIs) and improvement in sliding window algorithms would most likely translate to better performance when using the ROIs.

2.2.2.2 Combining the scored hypotheses

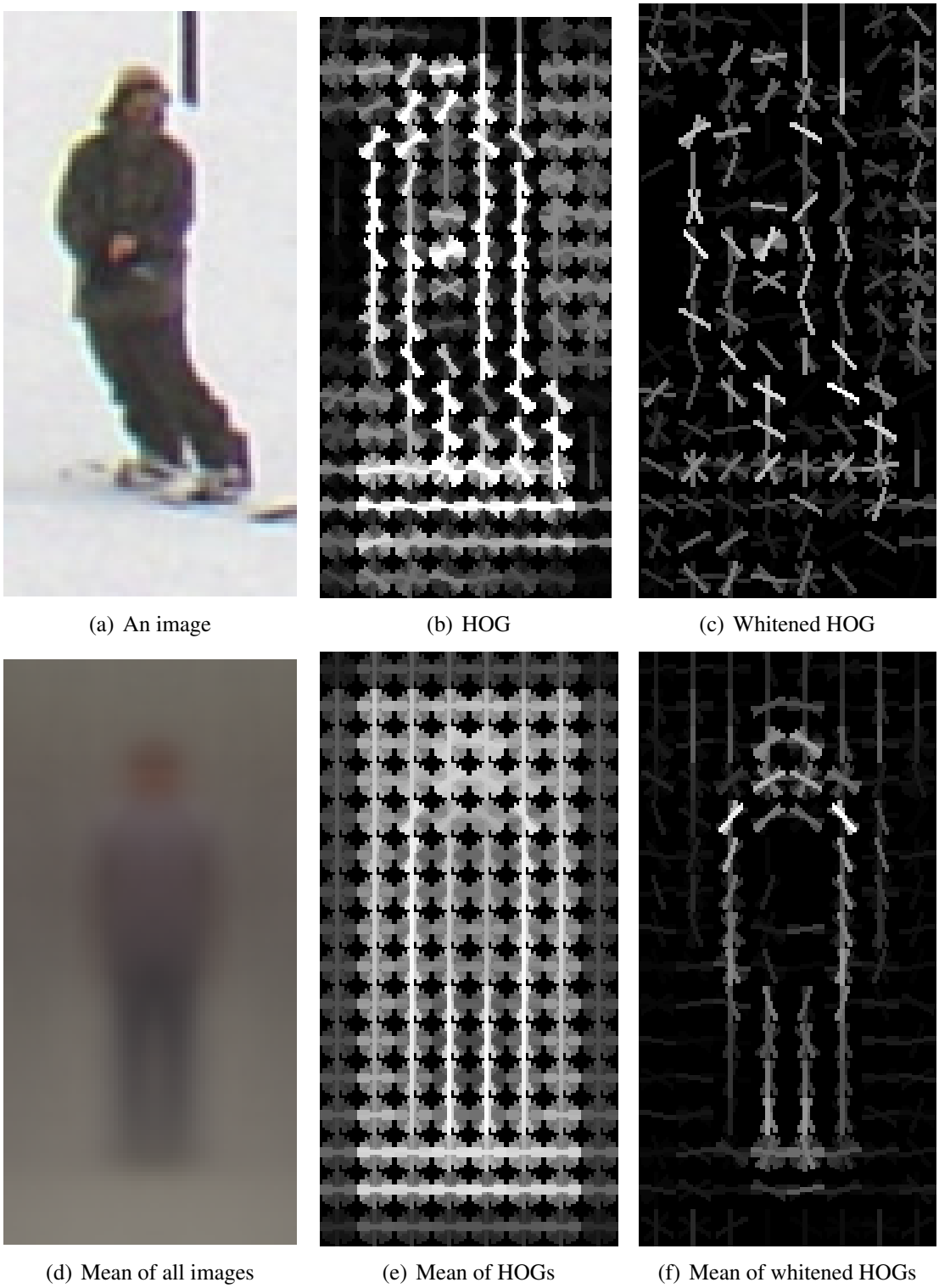
Multi-scale sliding-window approaches tend to produce overlapping scored hypotheses in various locations and scales of the image. These multiple responses need to resolved, merged where appropriate and spurious responses should be suppressed to produce a final set of window locations (*i.e.* bounding boxes). This is commonly known as *Non-maxima Suppression* (NMS). The two most popular approaches are using Meanshift clustering [19] and a greedy pairwise overlapping suppression approach [33].

2.3 Feature Extraction

We now very briefly describe Histograms of Oriented Gradients (HOG) [21], which is a mainstream feature extraction mechanism, as part of the sliding window detection approach. Given an image patch p, a simplified description of HOG is as follows:

1. The horizontal and vertical gradients, g_x and g_y respectively, of p are computed. These are known as *first order derivatives* of p.

2. From g_x and g_y, the gradient magnitude g_m and orientation g_o are calculated.



(a) An image (b) HOG (c) Whitened HOG

(d) Mean of all images (e) Mean of HOGs (f) Mean of whitened HOGs

Figure 2.4: Visualization of HOG features. (a) shows an image containing a person in the INRIA Person dataset. (b) illustrates the visualisation of HOG features of (a). (c) shows the HOG features of (a) after applying *whitening transform*. (d) gives the mean color image of all (positive) images in the INRIA dataset. (e) shows the visualisation of the mean of HOG features of all the images. (f) is the visualisation of the mean of whitened HOG features of all the images.

3. The image patch p is divided into a grid of non-overlapping 8×8 regions, called *cells*.

4. For each cell, a histogram of gradient orientations is built in that region using the corresponding values in g_o. Also when computing the histogram, the bins are weighted according to g_m in that region. This results in a histogram of orientations where stronger gradients (*i.e.* gradients with higher magnitude) are given higher weights than weaker gradients. The histogram is normalised by a suitable normalisation scheme such as dividing by the sum of the bin counts.

5. This process is repeated for all the cells in p. Then all these local histograms are concatenated to form one single feature vector $\mathbf{x}$.

As with HOG, most feature extraction mechanisms can be seen as a process of converting an image patch p to a feature vector $\mathbf{x}$. The HOG described above is a simplification of HOG proposed by Dalal and Triggs [21]. In [21], there are additional steps such as 4-way block normalisation (to increase illumination robustness) for each cell using the histograms in the cells nearby (called a *block*), tri-linear interpolation when counting bins of histograms and more sophisticated normalisation schemes. Figure 2.4 illustrates some visualisations of HOG features. In the figure, *whitening transform*[1] is used for better visualisation of HOG features. Given a 64×128 image patch p, 8×8 pixels sized cells, 9 orientation bins and 4-way block normalisation, the resulting feature vector $\mathbf{x}$ has the following length:

$$\mathbf{x} \in \mathbb{R}^k \quad \text{where } k = \frac{64}{8} \times \frac{128}{8} \times 9 \times 4 = 4608$$

2.4 Regularized Linear Classifiers

Regularized linear classifiers (RLCs) are a family of classifiers which includes many of the state-of-the-art classifiers commonly used for efficient large-scale learning. RLCs include linear Support Vector Machines (SVMs) [20], regularized logistic regression [51], Perceptron [36] and Winnow [67]. RLCs are becoming increasingly popular due to large

[1] Whitening transform is a process that removes linear correlations in the feature space. One of the ways of achieving this is by learning PCA on the HOG feature vectors, projecting the vectors onto the PCA space (to get PCA scores) and then dividing each dimension of the PCA scores with its standard deviation and then projecting back to the original HOG space. This transforms the covariance of the HOG features to the identity matrix and has the effect of removing the linear correlations and suppressing less interesting structures and thus allowing more interesting structures to become visible in the visualisations.

amounts of data that are available nowadays and fast sophisticated feature extraction algorithms that generate very high dimensional data for which linear classifiers are often sufficient and usually the only computationally feasible option [34,57]. Recently, explicit (non-linear) feature projection methods (*e.g.* [62,85,86,108]) are also becoming popular, further increasing the demand for RLCs.

Given training data $\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ where $\mathbf{x}_i \in \mathbb{R}^k$ is a feature vector and corresponding labels $\{y_1,\ldots,y_N\}$ where $y_i \in \{1,0\}$ is the supervision label associated with $\mathbf{x}_i$, most RLCs have the following form of optimization objective function:

$$\mathbf{m}_{\text{trained}} = \arg\min_{\mathbf{m}} \sum_{i=1}^{N} f_L(\mathbf{m},\mathbf{x}_i,y_i) + \alpha f_R(\mathbf{m}) \tag{2.1}$$

where $\mathbf{m}_{\text{trained}} \in \mathbb{R}^k$ is the vector of linear weights for the trained classifier. Furthermore, f_L is the *loss function*, f_R is the *regularization* function to penalize $\mathbf{m}$ of higher complexity and α is the trade-off term between the regularization term and loss term. For L2-regularization, we have $f_R = \mathbf{m}^T\mathbf{m}$ which penalises large values of $\mathbf{m}$ and for L1-regularization, $f_R = [1,\ldots,1]^T\mathbf{m}$ which encourages sparse solutions. For SVM,

$$f_L(\mathbf{m},\mathbf{x}_i,y_i) = \max(0, 1 - y_i\mathbf{m}^T\mathbf{x}_i) \tag{2.2}$$

and for logistic regression,

$$f_L(\mathbf{m},\mathbf{x}_i,y_i) = \log(1+\exp(-y_i\mathbf{m}^T\mathbf{x}_i)) \tag{2.3}$$

At test time, for SVM, the classifier score s_t of a test feature vector $\mathbf{x}$ is given by

$$s_t = (\mathbf{m}_{\text{trained}})^T\mathbf{x} \tag{2.4}$$

whereas for logistic regression, the score is given by

$$s_t = \frac{1}{1+\exp(-(\mathbf{m}_{\text{trained}})^T\mathbf{x})} \tag{2.5}$$

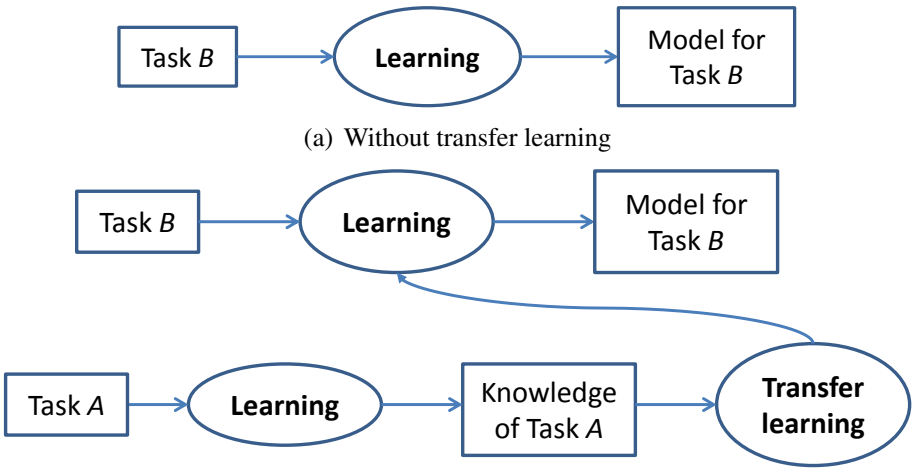
There are very efficient solutions for global optimization of equations of the form given in Equation 2.1 even for extremely large amounts of data with high dimensions. This makes training of RLCs efficient for large-scale learning [32]. Furthermore, as can be seen in Equations 2.4 and 2.5, prediction at test time involves mostly a single dot product using the learnt weights which can be performed very fast on modern machines.

Although the algorithms proposed in this thesis could be readily applied to most detection paradigms, for simplicity, only the sliding window paradigm is demonstrated.

Moreover, for simplicity and speed in feature extraction and learning, we use Histograms of Oriented Gradients (HOGs) and linear SVMs respectively, although any feature mechanism and classifier could potentially be used.

2.5 Transfer Learning

In this section, we give an introduction to transfer learning. Stated informally, the concept of transfer learning, in the field of machine learning, is mainly relevant when we have *related* tasks, and knowledge about some of those tasks; having knowledge about some tasks can be used to learn about other related tasks in an easier, faster or improved manner. This is useful because, for many tasks in machine learning, we may have a large amount of labelled data for a task A but may not have sufficient labelled data (or even no labelled data) for a task B which is related to task A in some way. Using transfer learning, we can transfer the knowledge that we have about task A to task B using some commonality between task A and task B. This is illustrated in Figure 2.5.



(a) Without transfer learning

(b) With transfer learning

Figure 2.5: Transfer learning between two tasks. Given two related tasks A and B, exploiting and transferring the knowledge about task A to task B can help learn a better model for task B.

Another useful benefit of transfer learning is when deploying trained models (*e.g.* classifiers) for prediction at *test* time in real-life systems. There is one assumption common to most machine learning algorithms: the distribution and the feature space of the training data are the same as those of the test data. Generally, if the *feature space* of the test data is different from that of the training data, the current model cannot be applied to the test data and a new model would have to be trained in the feature space that is the same as

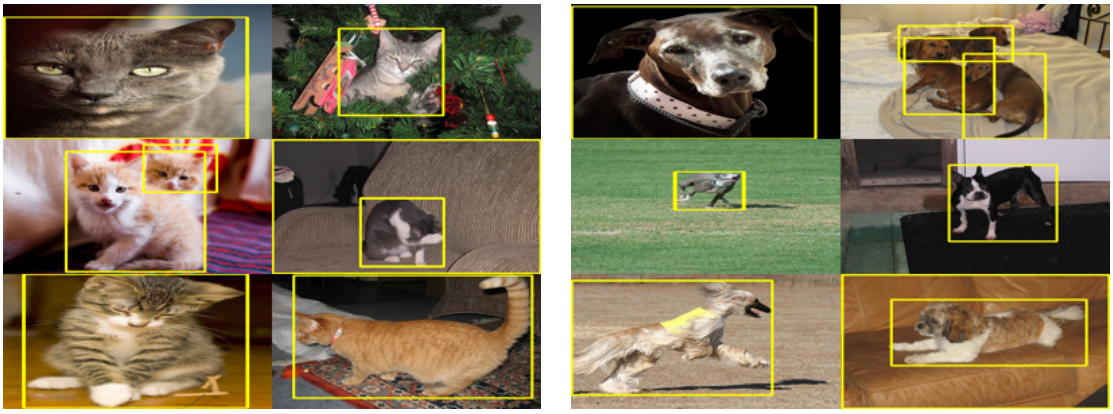
the test data. If the feature spaces of the training and the test data are the same but the *distributions* of the training and the test data are different, the model that was trained on the training data may perform poorly on the test data depending on the extent of the difference between the training and test data distributions. If this difference is large enough, the model might not even give any meaningful predictions and a new model would then need to be trained.

Having to train new models in this way can be computationally expensive and with traditional machine learning methods, this is usually necessary because for many deployed machine learning systems, the test data distributions are different from those of the training data. Transfer learning can help here to a certain extent by considering the training and test data as data for two related tasks.

We now discuss two commonly used terms in the transfer learning literature [80]: a *domain* and a *task*. Furthermore, when discussing transfer learning below, we would do it in the context of classification (since this thesis makes use of classifiers for pedestrian detection) although transfer learning can also be used for regression and density estimation.

A *domain* $\mathcal{D}$ is defined by a feature space $\mathcal{X}$ and a probability distribution $P(\mathbf{x})$ over the data associated with $\mathcal{D}$. A *task* $\mathcal{T}$ is specified by a label space $\mathcal{Y}$ and a distribution over the label space $P(y)$.

To give an example of a domain and a task, consider training a pedestrian classifier using the INRIA dataset. To simplify the explanation, assume that we are given cropped patches of pedestrians and non-pedestrians (also referred to as *positive patches* and *negative patches* respectively). For each patch, we extract features using any feature extraction algorithm to obtain a feature vector. The feature extraction mechanism defines the feature space $\mathcal{X}$. For instance, if we are using the HOG feature extraction algorithm which results in feature vectors of length 4608 and each dimension of a feature vector is a real number within the range of $[0, 0.2]$, then $\mathcal{X}$ is a 4608-dimensional space for which the values of each dimension has the range $[0, 0.2]$. After extracting features from each patch, we now have the training data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where the N number of training data are samples from the underlying distribution $P(\mathbf{x})$, *i.e.* $\mathbf{X} \sim P(\mathbf{x})$. For classification, each training datum $\mathbf{x}_i \in \mathbf{X}$ is also associated with a label y_i. There are N labels $\mathbf{Y} = \{y_1, \ldots, y_N\}$ for the training data. For pedestrian classification, the label is either *pedestrian* or *non-pedestrian*, *i.e.* $y_i \in \{\text{pedestrian}, \text{non-pedestrian}\}$. The training data $\mathbf{X}$ together with the labels $\mathbf{Y}$ is usually called a *labelled (training) dataset*. After obtaining the labelled dataset, a classifier can be trained using a supervised machine learning algorithm which produces a model that can be written as a function of $\mathbf{x}$: it takes in a feature vector $\mathbf{x}$ as input and produces a classification score as output. This can also be probabilistically interpreted



(a) Samples from cat dataset (b) Samples from dog dataset

Figure 2.6: Some samples of cats and dogs from the PASCAL VOC dataset [31]. Localisation of cats and dogs is shown with bounding boxes.

as $P(y|\mathbf{x})$, *i.e.* the (posterior) probability of the class labels y given a feature vector $\mathbf{x}$. In summary, the domain and the task for this pedestrian classification setting is given by $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$ and $\mathcal{T} = \{\mathcal{Y}, P(y)\}$ respectively.

	Source	Target
Domain	$\mathcal{X}_s, P(\mathbf{x}_s)$	$\mathcal{X}_t, P(\mathbf{x}_t)$
Task	$\mathcal{Y}_s, P(y_s)$	$\mathcal{Y}_t, P(y_t)$

Table 2.1: Annotation summary for transfer learning

The annotation summary for transfer learning is given in Table 2.1. Given a *source domain* $\mathcal{D}_s = \{\mathcal{X}_s, P(\mathbf{x}_s)\}$ and a *source task* $\mathcal{T}_s = \{\mathcal{Y}_s, P(y_s)\}$, the aim of transfer learning is to transfer the "knowledge" in $\mathcal{D}_s$ and $\mathcal{T}_s$ to a *target domain* $\mathcal{D}_t = \{\mathcal{X}_t, P(\mathbf{x}_t)\}$ and a *target task* $\mathcal{T}_t = \{\mathcal{Y}_t, P(y_t)\}$ so that the learning of $P(y_t|\mathbf{x}_t)$ is improved. Although multiple sources and targets can be used for transfer learning, in this thesis we focus only on one source and one target since this situation is prevalent in real-life situations.

As an example application of transfer learning, consider a scenario in which datasets of cats and dogs are given (shown in Figure 2.6) and the source task is the detection of cats and the target task is the detection of dogs. We would like to exploit the knowledge that we have about cats (assuming the availability of a large amount of labelled data for cats) and *transfer* it to the process of learning a dog classifier (assuming insufficient labelled dog data) to help obtain a better dog classifier. This is possible because even though detection of cats and detection of dogs are different tasks, they are related in that cats and dogs share some similarities in appearance, shape and structure (as can be seen in Figure 2.6). The feature spaces of cats and dogs, $\mathcal{X}_s$ and $\mathcal{X}_t$ respectively, may or may not

be the same, but $P(\mathbf{x}_s)$ and $P(\mathbf{x}_t)$ would be different. In addition, the label spaces of the source and target tasks, $\mathcal{Y}_s$ and $\mathcal{Y}_t$ respectively, are also different since $\mathcal{Y}_s = \{\text{cat}, \text{non-cat}\}$ and $\mathcal{Y}_t = \{\text{dog}, \text{non-dog}\}$ and $\mathcal{Y}_s \neq \mathcal{Y}_t$.

The example given above is a problem of *supervised transfer learning* because there is some labelled data available from the target dataset. An alternative setting is *unsupervised transfer learning* where there is no labelled data available from the target dataset.

2.6 Domain Adaptation

We now discuss a special case of transfer learning in which the source and target *tasks* are the *same* (*i.e.* $\mathcal{Y}_s = \mathcal{Y}_t$ and $P(y_s) = P(y_t)$) and the source and target *domains* are *different*. Moreover, even though the domains are different, the feature spaces of the source and target are the same (*i.e.* $\mathcal{X}_s = \mathcal{X}_s$ and $P(\mathbf{x}_s) \neq P(\mathbf{x}_t)$). This is known as *domain adaptation* which is actually a type of *transductive transfer learning* [80] and is simpler than the general transfer learning setting.

The reason for highlighting this particular form of transfer learning is that it can efficiently tackle the type of problem that we are interested in, which is adapting pedestrian detectors trained on a generic dataset (*e.g.* INRIA pedestrian dataset) to a specific scene (*e.g.* a surveillance video camera recording a traffic junction). This can be placed in a domain adaptation framework by assuming that the source domain is the data from the generic pedestrian dataset and the target domain is data that can be obtained from the specific scene. To show that this is a domain adaptation setting, the following observations can be made:

1. $\mathcal{X}_s = \mathcal{X}_t$: The source feature space is the same as the target feature space. This is because it is assumed that the same feature extraction mechanism is used for both the generic dataset and the specific scene.

2. $\mathcal{Y}_s = \mathcal{Y}_t$: The source label space is the same as the target label space. This is because for both the generic dataset and data from the specific scene, the label space is given by $\{\text{pedestrian}, \text{non-pedestrian}\}$.

3. $P(y_s) = P(y_t)$: The (prior) distributions on the labels for the generic dataset and specific scene are assumed equal.

4. $P(\mathbf{x}_s) \neq P(\mathbf{x}_t)$: The pedestrian distribution of the generic dataset is *not* the same as that of the specific scene. This is due to differences in image resolutions, illumination, pedestrian poses, camera angles, motion blur, *etc*. Even though $P(\mathbf{x}_s) \neq P(\mathbf{x}_t)$,

there is still some relation between $P(\mathbf{x}_s)$ and $P(\mathbf{x}_t)$, and $P(\mathbf{x}_t)$ can be considered as an (unknown) transformation of $P(\mathbf{x}_s)$.

As can be seen, this is exactly a domain adaptation setting where the tasks for the source and target are the same and domains are different. Many computer vision problems can be placed in this domain adaptation framework.

Another example scenario where domain adaptation may be helpful is when having a face detector that is trained using a generic face dataset (such as the Faces in the Wild dataset [52] which contains over ten thousand images of faces collected from the Internet) and wanting to apply the detector to images taken in a more specific and controlled environment (such as the Yale Face Database [63]). Random samples from these datasets are shown in Figure 2.7. Although the face detector trained on the generic dataset may work reasonably well on the target dataset, it is expected that adapting the detector to specialise it to the target dataset (which may have much less intra-class variation of faces) might improve the detection performance in the target dataset. This is a domain adaptation problem because the feature spaces of the source and target datasets are the same since faces are represented by the same feature extraction mechanism (such as HOG or Haar features). Moreover, the tasks are the same since the label spaces and the prior label probabilisties are the same (both aims at face/non-face classification).

As with transfer learning, there are two main types of domain adaptation. In both types, we assume that we have a sufficiently large number of labelled data for the source dataset. The first type is *unsupervised domain adaptation*. In this type, we do not have any labelled data in the target dataset. In the second type, we assume that we have some labelled data in the target domain. This is known as *supervised domain adaptation*. In this thesis, we are concerned with only unsupervised domain adaptation.

2.6.1 Domain Adaptation vs. Semi-supervised Learning

In this subsection, we digress to briefly compare domain adaptation with the more well-known semi-supervised learning. Unlike domain adaptation, semi-supervised learning does *not* have the notion of source and target datasets and assumes that the distributions of training and test data are the same. In this sense, semi-supervised learning is equal to standard supervised machine learning except that in semi-supervised learning, the majority of training data are *unlabelled*. But both the labelled data and unlabelled data are still assumed to come from the same distribution.

Formally, in semi-supervised learning, the dataset can be written as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{Y} = \{y_1, \ldots, y_N\}$ and most of $y_i \in \mathbf{Y}$ are unknown and can be considered as *latent* vari-

(a) Face samples from Faces in the Wild dataset [52] (b) Face samples from Yale Face Database [63]

Figure 2.7: Samples of faces from source and target domains. Note that for Yale dataset (on the right), only greyscale images could be obtained. Therefore, both of these datasets are shown in grayscale.

ables. Despite the difference between domain adaptation and semi-supervised learning, some semi-supervised learning techniques can be adopted for *certain* domain adaptation problems. This can be done by treating the data for which labels are known as the source dataset and the data with the latent labels as the target dataset. This however violates the assumption made in semi-supervised learning that each data $\mathbf{x}_i \in \mathbf{X}$ (regardless of whether its label is known or latent) must come from the same distribution. However, for certain situations such as where the difference between the source and target datasets is small, formulating the domain adaptation in such a way might be acceptable.

2.7 Weakly Supervised Learning

Weakly supervised learning is a machine learning paradigm distinct from supervised or semi-supervised learning. In weakly supervised learning, as the name suggests, the training data are only *weakly* annotated. For example, rather than being given exact patches of objects, the algorithm may be given only weak indications of the presence of objects in images.

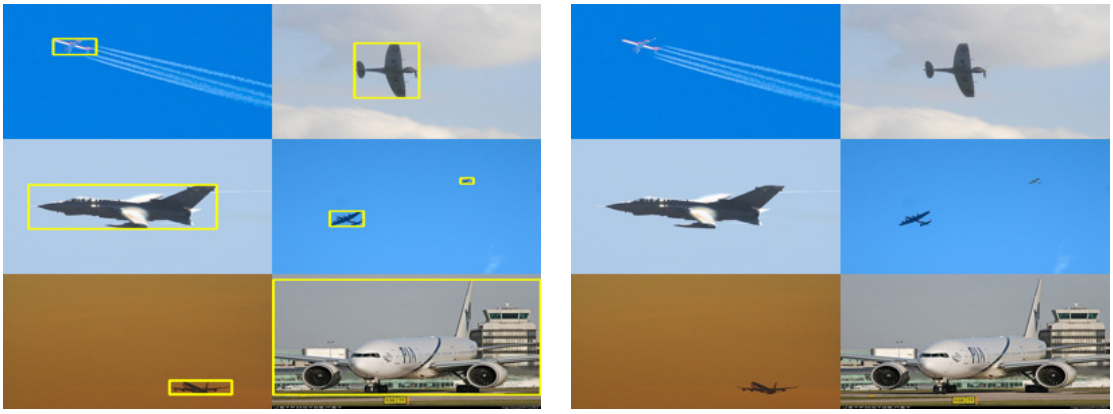
The weakly supervised learning problem (for binary classification) can be formalised as follows. Let the training dataset be made up of k groups (or *bags*):

$$\{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_k, y_k)\}$$

where $y_i \in \{1, 0\}$. A bag $\mathbf{X}_i$ consists of several data *instances*, *i.e.* $\mathbf{X}_i = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots\}$ where $\mathbf{x}_{ij}$ is a feature vector belonging to either the positive or negative class (for binary classification). However, the labels of the instances are *not* observed (*i.e.* they are latent variables). Instead, the entire bag is associated with a *single* bag label which can be either a *positive bag* if $y_i = 1$ or a *negative bag* if $y_i = 0$. Each positive bag contains *at least* one positive instance and each negative bag contains *only* negative instances.

In order to make it more general, although labels are not given for each instance, in each positive bag, the probability distribution of the instances belonging to the positive class may be given, *i.e.* in a positive bag, some instances may be more likely to be from the positive class than others. If no such (prior) information is available, then a uniform distribution is assumed, *i.e.* in a positive bag, all the instances are equally likely to be from the positive class.

As an example of weakly supervised training, consider training an aeroplane detector. Figure 2.8 illustrates a comparison between annotation requirements for standard supervision and weak supervision in the context of training the aeroplane detector. Instead of



(a) Standard supervision (b) An example of weak supervision

Figure 2.8: An example of the comparison between standard supervision and weak supervision. Weak supervision requires less human effort than standard ("strong") supervision. Note that negative images (*i.e.* images that do not contain any aeroplanes) are not shown because they are the same for both supervised and weakly-supervised learning cases and they are relatively cheap to obtain.

being given patches corresponding to aeroplanes (*i.e.* bounding boxes of aeroplanes in the training dataset), we may only be given images containing aeroplanes without any information about the exact location and extent of aeroplanes in each image; the only cue that is given is that each training image contains one or more aeroplanes. This is a weak label as opposed to a strong label in the form of exact bounding boxes of aeroplanes.

In the context of the formalisation described previously, in this case, each training image (regardless of whether the image contains an aeroplane or not) can be considered as a bag and each image shown in Figure 2.8(a) can be considered a positive bag. Each positive bag would then consist of a set of feature vectors corresponding to all the possible patches (*i.e.* instances) at various locations and scales of the corresponding image. However, in a positive bag, only one of those patches would truly be the patch corresponding to an aeroplane but this information is not known. As can be observed, without any prior information about which patches are more likely to be positive instances in each bag, it is very computationally expensive to jointly infer where an aeroplane is in each positive training image from the space of all possible patches in all the training images.

Instead of weak supervision at the level of objects, we may also have weak supervision at the level of object parts. An example of this is the Deformable Part Models [33].

It should be noted that weakly-supervised learning is different from semi-supervised learning in that in semi-supervised learning, although we have a small number of labelled data (and a lot of unlabelled data), for the labelled data that is available, the supervision is *not* weak.

Chapter 3

Literature review

3.1 Introduction

Domain adaptation is a relatively new research area. Early works on domain adaptation were published in the field of text and Natural Language Processing (NLP) [8, 53, 71, 88].

Hwa [53] proposes an adaptation approach for grammar structure induction using sparsely annotated training data (*i.e*. data with limited constituent information) to obtain results that are almost as good as using a fully annotated textual corpus. Roark and Bacchian [88] make use of a maximum a posterior framework to adapt probabilistic context-free grammars to new domains. McClosky *et al*. [71] propose a parser adaptation system using self-training and re-ranking. Blitzer *et al*. [8] propose an unsupervised domain adaptation for part-of-speech tagging by projecting the source dataset to a real-valued low-dimensional feature representation that is shared across the source and the target domains. This representation is learnt using structural correspondence learning which works by firstly defining a set of pivot features. Pivot features are frequently-occurring features that are invariant and discriminative across both domains. Secondly, correspondences among features of source and target domains are learnt with the help of these pivot features. Their proposed algorithm assumes that features from the domains are binary and also requires defining pivot features, which is not trivial especially in applications other than text.

In fact, most of the algorithms used for domain adaptation for NLP are not suitable for

vision applications. Therefore, in this chapter, we will focus on prior work about domain adaptation for computer vision rather than NLP.

Research for domain adaptation for vision is even more recent than NLP. There are mainly two areas of research in domain adaptation for vision: image classification and object detection. For image classification, the majority of the approaches for domain adaptation turn out to be metric learning or feature projection approaches.

Object detection is a harder and a more general task than image classification. Similarly, domain adaptation for object detection is generally a more challenging problem than domain adaptation for image classification. Some of these challenges are:

1. Extreme class imbalance. Object detection involves having to model the positive and negative class. The number of data in the negative class is much larger than that of the positive class and the positive class can easily get swamped with the negative class.

2. The adaptation algorithm not only has to deal with the intra-class variation of the positive class but also the much larger "sea" of intra-class variation of the negative class.

3. Due to object detection having different requirements as compared to image classification (such as needing to evaluate hundreds of thousands of candidate windows), object detection systems usually use a different set of features (such as Histogram of Oriented Gradients or Haar features) than image classification systems (which tend to use features such as "Bag of Visual Words"). Object detectors typically use very high dimensional and dense features whereas many image classification systems tend to use lower-dimensional and sparser features. This makes a difference in the required domain adaptation techniques. For example, generative models may be suitable for domain adaptation for image classification but ill-suited for domain adaptation for object detection.

It is therefore *not* straightforward or trivial to apply existing domain adaptation methods for image classification to the task of object detection. Furthermore, domain adaptation for object detection in *videos* brings with it a unique set of challenges and opportunities which is different from that of image classification or even object detection in static images. Some of these opportunities include availability of spatio-temporal smoothness and other types of information that can be learnt and exploited from videos. This means that even if adopting existing domain adaptation techniques for image classification for object detection is easy, it may be more desirable to research and develop algorithms that

exploit these cues in the videos for improved performance. Moreover, for far-field videos, which we tackle in this thesis, pedestrians are of small resolution which further increases the challenge of domain adaptation. Therefore it is crucial to differentiate domain adaptation approaches for image classification from those for object detection (especially in videos).

Due to its unique challenges and opportunities, it turns out that different variations of *self-training* is the most popular approach for state-of-the-art domain adaptation for object detection in videos. The popularity is due to the fact that the self-training framework is flexible, can work with a variety of discriminative classifiers and can incorporate different types of prior knowledge in a natural and easy way. For object detection, we will mainly focus on domain adaptation of object detectors trained on image datasets to videos.

To motivate the structure of the literature review, we first recap the main contributions in this thesis. The first contribution is domain adaptation using state-of-the-art deep feature learning, the second is about making domain adaptation easy by utilising cues in videos, and the third is about weakly-supervised learning. Although all these three contributions have the same underlying theme of reducing the manual annotation effort, the first and second contributions are much more related to each other (since they are both about domain adaptation); therefore we review the related work for these two contributions in Section 3.2 and we review the work related to weakly supervised learning separately in Section 3.3. Moreover, we put into context and relate the three contributions made in this thesis to state-of-the-art research in Section 3.4.

3.2 Domain Adaptation

For the literature review on domain adaptation, we begin by discussing research related to domain adaptation for image classification (Section 3.2.1). Then we review domain adaptation for object detection for videos in Section 3.2.2. For the sake of completeness, in Section 3.2.3, this is followed by reviewing two areas that are not directly relevant but somewhat related to our thesis:

- Learning moving object detectors in videos (Section 3.2.3.1): In this section, we discuss approaches that learn class-agnostic moving object detectors in videos.
- Semi-supervised learning for object detection in videos (Section 3.2.3.2): Here, we review algorithms that learn object detectors in videos using semi-supervised learning (given a small amount of labelled data in the target domain).

Figure 3.1: The need for domain adaptation for image classification. Figure taken from [94].

3.2.1 Domain Adaptation for Image Classification

The need for domain adaptation for image classification is illustrated in Figure 3.1. Most of the research in this area is based on learning a common feature representation across the source and target domains.

One of the earliest domain adaptation approaches for image classification is the work by Saenko *et al*. [94]. They provide a supervised domain adaptation algorithm that learns a regularised non-linear transformation that is invariant across the source and target domains. Learning such a transformation allows modelling of changes resulting from the difference in the source and target domains. They also introduce a multi-domain object database (shown in Figure 3.2) to evaluate domain adaptation algorithms for image classification. Their method requires exact manual mapping of samples from the source and target domains which can be very time-consuming.

An extension of [94] is proposed by Kulis *et al*. [60]. This time, instead of learning a single transformation as in [94], the authors propose learning asymmetric linear transforms, *i.e*. two linear transformations: one for the source domain and the other for the target domain, to respectively project the source and target data to a common subspace. In order to deal with non-linear asymmetric transformations, they kernalize the algorithm (by running the algorithm in the kernel space instead of the original feature space). Their approach however shares the same limitation as [94]: they require manual specification of pairs of source and target data examples that are similar semantically (*e.g*. two very similar cups (or even the same cup) taken from the source and target domains may form such a pair).

Gopolan *et al*. [45] propose an unsupervised domain adaptation method that models the domain shift by gradual changes in the representation from the source to target

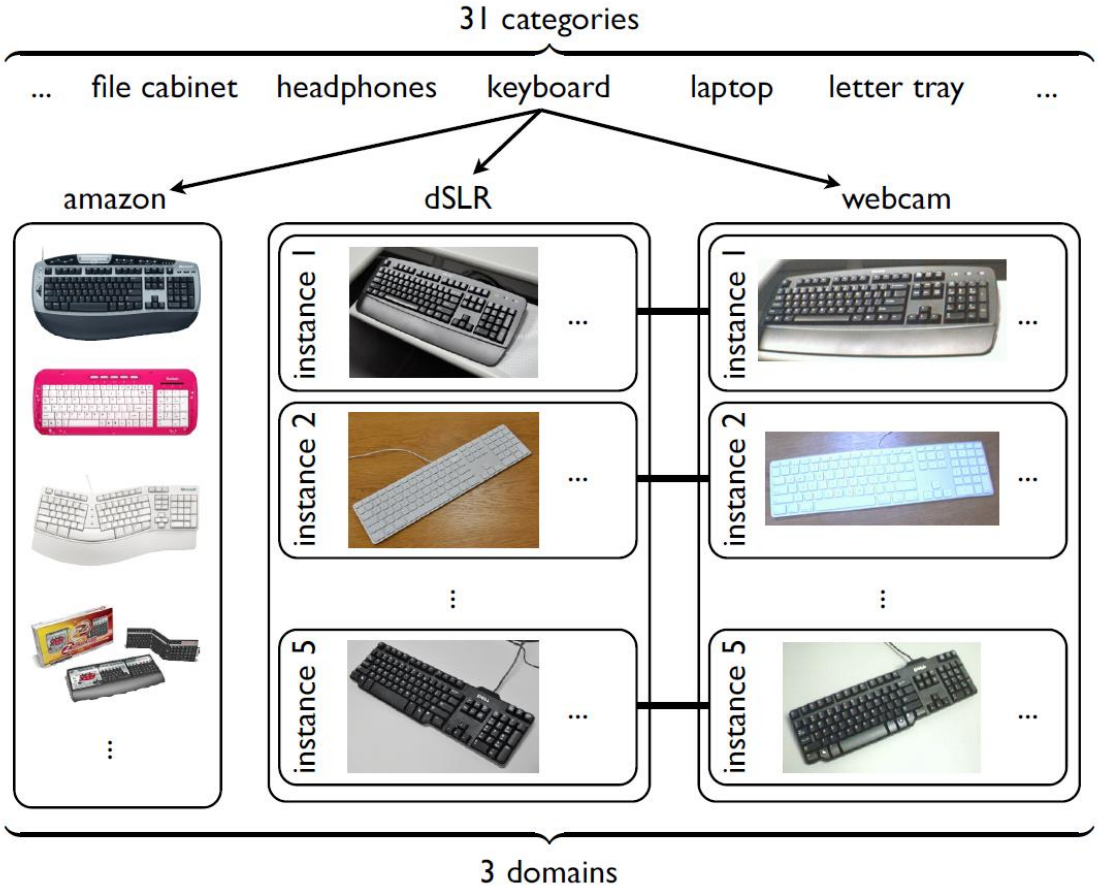


Figure 3.2: Multi-domain object database to study and evaluate domain adaptation algorithms for image classification proposed by [94]. The database contains 31 object categories and for each category, there are 3 domains: images taken from `Amazon.com`, a high resolution digital SLR camera and a simple low resolution webcam. Figure taken from [94].

domain. This is achieved by modelling the subspaces in the source and target domains and then generating intermediate subspaces between them as sampled points along the geodesic on the Grassmann manifold [72]. This is shown in Figure 3.3. Their approach requires tuning of many parameters including determining the finite number of subspaces to sample.

Gong *et al*. [44] present a system similar to [45]. They propose a method called "geodesic flow kernel" (illustrated in Figure 3.4) which is an improvement on [45] in that it eliminates the need to sample a finite number of subspaces and to tune as many parameters as [45] by "kernalizing" the approach of [45] and considering an infinite number of subspaces.

Mirrashed and Rastergari [74] approach unsupervised domain adaptation by learning

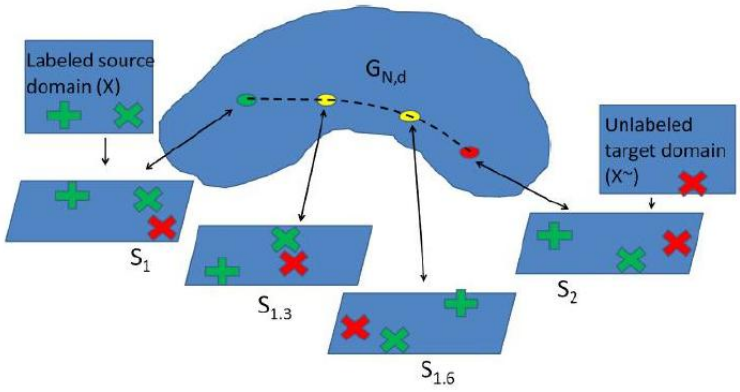


Figure 3.3: Illustration of sampling points between the subspaces of the source and target domains on the Grassmann manifold. In the figure, the Grassmann manifold is represented by $G_{N,d}$ which is basically the space of d-dimensional subspaces in $\mathbb{R}^N$ and S_1 and S_2 are two points on $G_{N,d}$ corresponding to the source and target domains respectively. The ones in between S_1 and S_2 can be considered as intermediate subspaces (*i.e.* intermediate points on the Grassman manifold) going from the source point to the target point. Figure taken from [45].

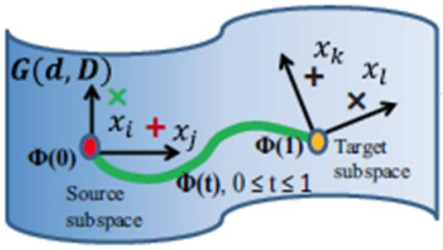


Figure 3.4: Geodesic flow kernel to model the gradual change from the source domain, $\Phi(0)$, to the target domain, $\Phi(1)$. $\Phi(t)$ gives the subspace at any point, *i.e.* $0 \leq t \leq 1$, along the geodesic. Figure taken from [44].

a set of discriminative and invariant feature projections (into binary space) that models the class structures across source and target domains. Each of these projections is essentially a hyperplane in the feature space and a binary "attribute" is obtained by looking at which side of the hyperplane the data falls on. Using this set of projections (*i.e.* hyperplanes), the source dataset is projected into the binary space to get the binary attributes and the target classifier is obtained by training a classifier using the projected data. There is however no proper justification as to why the space should be binary in the first place and the proposed optimization algorithm is prone to local optima.

3.2.2 Domain Adaptation for Object Detection in Videos

Bose and Grimson [10] propose an unsupervised domain adaptation system for adapting a (baseline) detector trained on a far-field video (or a set of far-field videos) towards a different far-field video by a two-step self-training algorithm. In the first step, the baseline detector is used to score and label the unlabelled data (*i.e*. all sliding windows of frames in the video). A new classifier is then trained on the combination of the original data (from which the baseline detector was trained) and the most confidently scored data of the unlabelled data. In the second step, this newly trained detector is applied to the video and scene-specific features (*e.g*. silhouette height obtained by background subtraction) are extracted from the detections and a new classifier is trained with these features. There are a few limitations with this domain adaptation approach:

1. There is a need to determine the threshold for the "most confident" detections.

2. It is not known for sure whether the classifier obtained at the end of the first step is good enough. If it is not, then the second step will carry on the errors and may even make it worse. In other words, the second step is completely dependent on the outcome of the first step and has no chance of correcting any errors of the first step.

3. The final detector (at test time) is (still) dependent upon the results of background subtraction (in order to extract the scene-specific features). This can be a problem if the background subtraction is very noisy, especially for complex and cluttered scenes.

In addition, in the paper, it is not clear whether the performance improvement comes from the actual detector adaptation or from using a better (*i.e*. higher level) feature extraction mechanism.

A system that adapts a set of general part detectors to specific video scenes is proposed by Wu and Nevatia [115]. This is achieved by using a self-training framework where the "oracle"[1] is the (global) combination of the part detections; the global shape model given by the configuration of parts provides an additional and complementary source of information compared to the (local) part detectors. The approach is limited to boosting-type classifiers and to object detection systems that explicitly model objects with parts.

[1]The "oracle" is the verification process that selects which examples to include for each self-training iteration. It is often the most important component of a self-training algorithm. In order to maximise the efficiency and effectiveness of the self-training process and to minimise drifting, the oracle should be as independent as possible from the original (*i.e*. source) dataset and should offer complementary information to the information already contained in the source dataset.

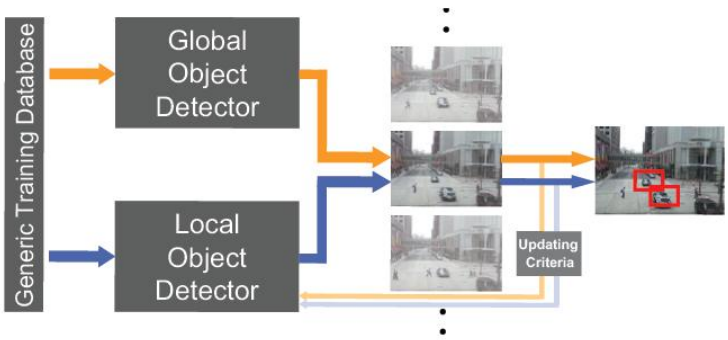


Figure 3.5: Detector adaptation approach of Kemhavi *et al*. [58] by combining the predictions of a fixed global detector and an online updated local detector. Figure taken from [58].

A Multiple Kernel Learning based self-training algorithm is used by Kemhavi *et al*. [58] to tune a generic vehicle detector to a traffic intersection. Their adapted detector is a combination of two separate detectors: one is termed a "global detector" which is the detector trained on the generic dataset and fixed (*i.e*. no updates are performed), and the other is an online detector updated with a simple self-training approach: most confident positive and negative examples scored by the global detector are added in each round. This is shown in Figure 3.5. For adding negative examples, examples are added that are both confident and have high positional entropy relative to the positions (in the image plane) of the currently collected negative image patches. This is to prevent too many negative patches from the same background position from being added. The process of using the global detector as an oracle in this way may not be very effective because the online classifier may never get better if the global detector (which is fixed) does not perform very well in the first place and additionally, the global detector does not provide any new complementary information to the online detector (since the online classifier is obtained from the global detector). Moreover, their method only applies to a particular type of classifier (*i.e*. Multiple Kernel Learning). Another potential problem is due to the final classifier being the combination of the global detector and the online classifier: there is a limit to the amount of adaptation the final classifier can undergo. For example, if the generic detector has a lot of false positives, it would still influence the final classifier to a large extent. And finally it is non-trivial to manually specify the best combination of the global detector and the online detector.

Wang *et al*. [113] propose a self-training algorithm to adapt a generic pedestrian detector to a specific scene. Their algorithm does not utilise background subtraction or (explicit) object tracking and it works as follows. Firstly the detector is applied on frames of

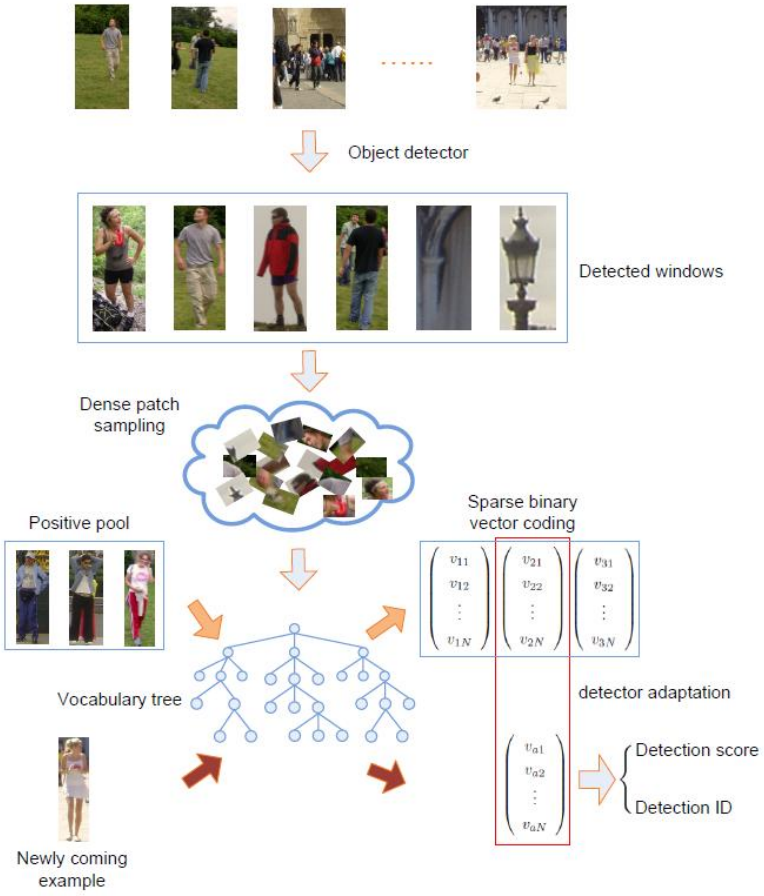


Figure 3.6: System overview of the algorithm of Wang *et al*. [113]. Figure taken from [113].

the video with a high recall and low precision setting. Then a hierarchical k-means tree is constructed using the features of these detections. Thirdly, the most positive and negative detections are identified and they are encoded using the learnt tree to obtain binary codes and a classifier is trained on this binary feature space. This is the scene-specific detector (illustrated in Figure 3.6). The performance is sensitive to setting and manual tuning of many parameters such as choosing a suitable low precision and high recall setting, thresholds for collecting confident positive and negative examples, the depth of the hierarchical k-means clustering and parameters for similarity measure for the binary features. It is uncertain whether the improvement of the scene-specific detector over the generic detector comes from the adaptation stage or the non-linear feature encoding stage (two unrelated steps). Furthermore, it is highly likely for the first step to fail to collect sufficient labelled data (due to collecting only the most confident positive and negative examples) to train a scene-specific detector that has good generalisation properties in the target domain.

Another self-training method is proposed by Sharma *et al*. [99] to adapt a generic detector to specific scenes for the task of pedestrian detection. The classifier used is Real Adaboost [37] with Multiple Instance Learning [23] loss function. The self-training approach works by applying the current detector to frames and then associating the detections into tracks. Then successfully tracked detections are added as new positive examples, and detections that do not belong to any of the tracks, are considered as new negative examples. For each detection to be added as a positive example, instead of directly adding the patch corresponding to the detection, the original patch and multiple patches surrounding the patch are treated as samples in a positive bag with the assumption that one of these patches contain the correctly localised positive example (as in the standard Multiple Instance Learning framework) . This is used to reduce the patch alignment errors commonly associated with collecting examples from detections. Their approach however is limited to Real Adaboost classifiers and the datasets that they are evaluating their algorithms on are high resolution datasets and only one type of moving object (*i.e*. pedestrian) is present in the scene. The Multiple Instance Learning approach that they have adopted is not likely to work well for low resolution videos such as the far-field surveillance scenes that are used for the experiments presented in this thesis.

Tang *et al*. [59] adapts a detector trained on an image dataset to a video based on a variation of iterative self-training which they term "self-paced domain adaptation". It works by adding easiest examples to the dataset first followed by increasingly more challenging ones. However, the self-paced domain adaptation technique is not much different from the traditional self-training approaches which seek to iteratively add the most confident detections in each round to *slowly* adapt the classifier to minimise the risk of drifting. For selecting examples to add in each round, instead of scoring individual detections, they score tracks in order to average out noise associated with individual detections. They assume that negative examples are known in the scene which means that their approach requires partial supervision.

To adapt a face detector in the form of a pre-trained cascade of classifiers to a new domain, Jain and Farfade [54] use a supervised domain adaptation algorithm. Their approach is essentially a type of self-training method where the oracle is a generative appearance model. They tested their algorithm by adapting a generic frontal face detector (such as the one available in the OpenCV library) to images containing baby faces. The approach however is limited to classifier cascades, requires a few hundreds of labelled annotation in the target domain and therefore is labour-intensive.

Sharma and Nevatia [100] present a self-training approach to adapt a pedestrian detector to video scenes. In order to collect samples for self-training, they apply the baseline

detector and keep only the most confident detections. Then the detections are placed into tracks using appearance, size and position cues. After the samples are collected, the positive examples are divided into different subcategories by applying a pre-trained pose classifier. Then they train a random fern classifier [78] for each positive subcategory to increase the precision of the baseline detector. From the evaluation in [100], it is not clear whether the detection improvement comes from the subcategory division and training nonlinear random fern classifiers or the actual adaptation algorithm itself. Moreover, the method requires a pose classifier for pedestrians to be trained and also involves non-trivial tuning of multiple parameters such as the thresholds for applying the detector in "high precision setting" for collecting samples during the adaptation stage and "high recall setting" at test time. Lastly, the adapted algorithm only reduces false positives and does not increase recall.

A co-training approach is adopted by Mirrashed *et al*. [73] to adapt vehicle detectors from multiple source domains to a target domain. Classifiers trained on different source domains iteratively train and improve each other by teaching, in each iteration, the most confident detections of one classifier to the other classifier(s). The algorithm makes use of Transfer Component Analysis [79] in order to reduce the effects of domain shifts between the datasets. As with other iterative self-training algorithms, the algorithm requires setting the threshold for selecting confident detections. Moreover, the system requires the use of multiple source domains which may not be feasible in many situations.

Shu *et al*. [101] propose a self-training approach to adapt a generic pedestrian to specific videos. Firstly, the generic detector is applied on frames and then the most confident detections are collected as positive examples. Negative examples are collected from the scene background. Then super-pixels are extracted and patches corresponding to the super-pixels are clustered to form a visual dictionary. This is used to encode the examples using a Bag of Words (BOW) approach. Then a classifier is trained using the encoded examples. Next, the classifier is applied on frames and this process repeats until convergence. The approach also requires setting of several sensitive hyper-parameters such as the parameters of super-pixel generation, the number of clusters for building the dictionary and the confidence threshold for positive sample collection. Since the negative examples come only from the scene background, the algorithm may not work well for videos where there are multiple moving objects. The evaluation is performed only on quite straightforward datasets where there are only pedestrians moving about. Furthermore, the super-pixel extraction may not work well for videos where pedestrians are medium or small-sized. Most importantly, it is again not clear whether the adaptation performance actually comes from the adaptation algorithm or from using a better feature

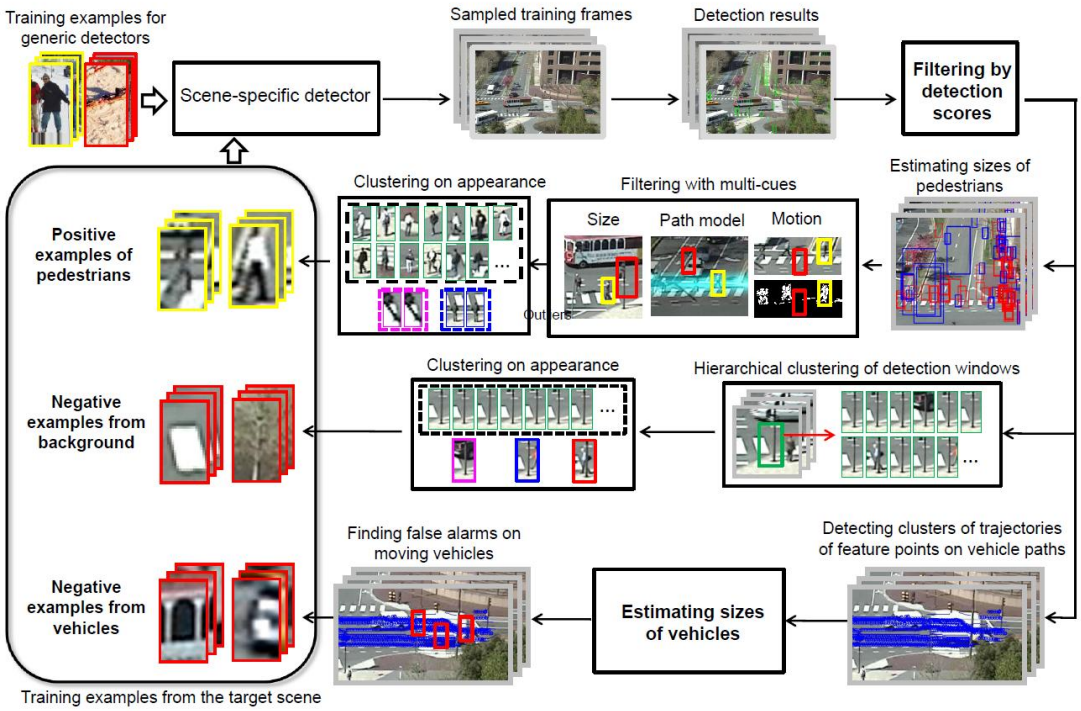


Figure 3.7: An iterative self-training technique of Wang and Wang [111]. In each iteration, positive and negative examples are collected by filtering with a variety of cues, added to the current dataset and a new classifier is trained. Figure taken from [111].

extraction mechanism than the baseline detector. This is important because if the baseline classifier uses the same feature extraction mechanism (*i.e.* super-pixel generation and BOW encoding) then it may be as good as the final classifier. If this is the case, then it would imply that the major part of the novelty is not in detector adaptation but in feature engineering.

The method proposed by Wang and Wang [111] iteratively improves a generic pedestrian detector by selecting new confident examples to add to the current dataset for retraining at every iteration. In order to collect examples for each self-training iteration, their oracle is a combination of vehicle and pedestrian paths, multiple different cues such as bounding box locations and sizes, background subtraction, thresholds, filters and hierarchical clustering. To obtain vehicle and pedestrian paths, they use the method of [112] which discovers motion patterns in long-term videos using topic models such as Hierarchical Dirichlet Processes [104]. The motion patterns are discovered in a bottom-up manner by treating quantized optical flow velocity and position (in the image plane) as low-level features, small video clips as documents and then co-clustering using the topic models. The approach requires quite extensive parameter setting and tuning such as deciding the length of a video segment (for topic modelling), setting the hyper-parameters

for optimizing the topic model and determining various parameters for different filtering steps, clustering and background subtraction and thresholds for object sizes. There is also a need to manually label the discovered paths and an assumption that pedestrians and vehicle paths are not overlapped to a certain degree. Lastly, the number of iterations for self-training is also required to be set and there is a possibility of drifting if too many iterations are performed. The overview of their system is shown in Figure 3.7. The method is extended in [110] by incorporating techniques such as reweighting the source data, confidence propagation and using the confidence when retraining rather than hard thresholding.

3.2.3 Areas Related to Domain Adaptation

We now describe two areas of study somewhat related to domain adaptation for object detection in videos. Firstly, in Section 3.2.3.1, we highlight research on learning general moving object detectors in video. Secondly, in Section 3.2.3.2, we discuss semi-supervised learning of object detectors in video.

3.2.3.1 Learning moving object detectors

The most common way of detecting foreground objects in video is to use background subtraction followed by a grouping technique such as Connected Component Analysis [95]. For more information on different approaches to background subtraction, the reader is referred to various surveys [13, 17, 83].

In this subsection, we focus on approaches based on training (*i.e*. learning) *classifiers* to model and detect general foregrounds (*i.e*. significant objects) in the scene, often improving the results of traditional background subtraction approaches by utilising the *generalising* (and noise-reduction) power afforded by the classifier training stage. The research problem tackled by these approaches is different from domain adaptation explored in this thesis. Moreover, their goal is to "blindly" detect *any* foreground object in the scene as opposed to being *aware* of specific object classes and detecting them in the scene. However, we review these papers for the sake of completeness since some of these methods do use self-training-like algorithms.

Nair and Clark [75] propose an approach for online learning of a moving object detector for an office corridor scene. An online Winndow classifier is trained on features extracted from foreground blobs obtained by background subtraction if foreground blobs have the correct aspect ratio and size corresponding to pedestrians. They evaluate their approach only on indoor scenes (shown in Figure 3.8) where there is only one type of

Figure 3.8: Office corridor scene used in [75]. Figure taken from [75].

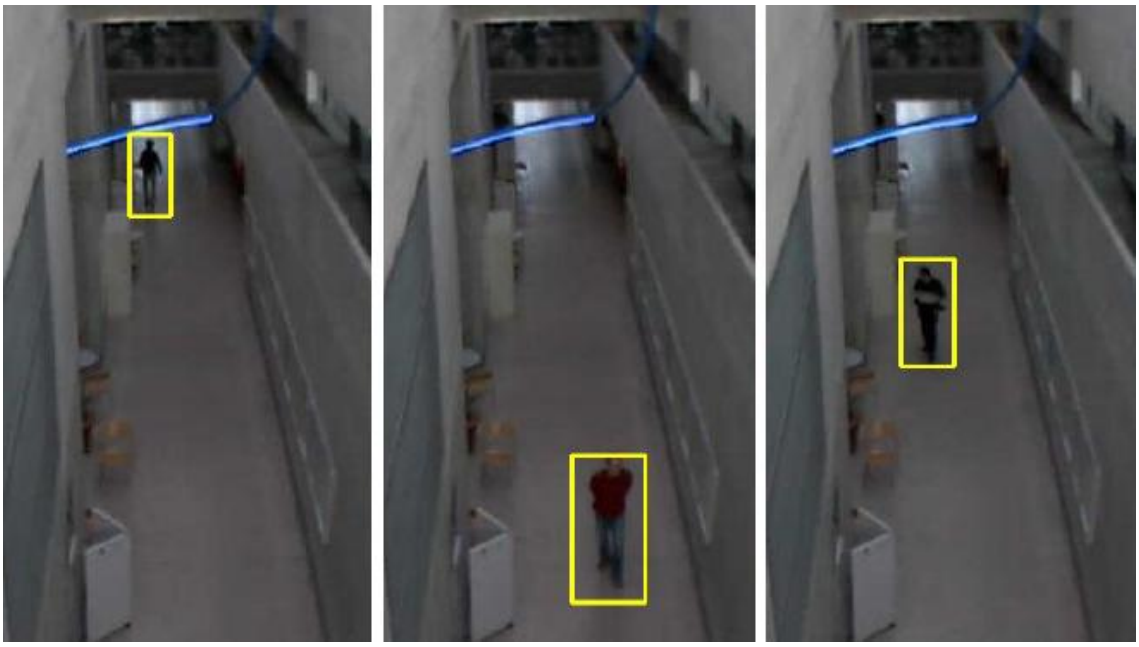


Figure 3.9: Indoor scene for pedestrian detection in [46]. Figure taken from [46].

moving object (*i.e*. pedestrian) for which background subtraction already performs quite well due to the restricted environment, where there are no major problems such as background clutter, multiple categories of objects, large illumination variation and large cast shadows (which would change the aspect ratios and sizes of detected blobs) that would be common in a lot of outdoor surveillance type scenarios. The (intra-class variation of) background clutter of their indoor scene is quite small, making the dataset not particularly challenging. A similar system using online Adaboost is proposed by Roth *et al*. [90].

Grabner *et al*. [46] propose a "grid-based" pedestrian detection[2] system based on

[2]As will be explained later, their approach actually reduces to a foreground detection system in scenes where there is more than one category of moving objects. This is why we discuss their approach here.

training one classifier for each image location (in the form of a pedestrian-sized window) and updating them independently online based on a simple update heuristic. The update strategy works as follows: they fix the positive class (with a small number of pedestrian examples) for all the classifiers without any update and *always* update the negatives with the assumption that the probability of wrongly updating the negatives is very small. The method assumes that the intra-class variation of the negative class (*i.e*. non-pedestrian patches) at each image location is extremely small and takes advantage of this to simplify the complexity of each classifier. While this may be the case for some scenes, it is not true for many scenes especially those where there is more than one class of objects. In those types of scenes, many image locations would still have to handle large intra-class variations (given by the combination of intra-class variations of the background at that image location and other object categories that may occupy the image location at any time), rendering the original intention of simplifying the task of the classifier ineffective. There are a number of additional potential problems associated with the approach:

1. The positive class is fixed and never updated, which means that the system may never detect some pedestrians which are not well represented by the initial set of pedestrian examples.

2. The negative class is always updated, which means that the negative class of each classifier will be dominated by the background of the image location corresponding to the classifier. This means that other (*i.e*. non-pedestrian) classes of objects that occasionally move inside the image location would most likely be erroneously classified as "pedestrian" since the classifier will be quite certain that it does not belong to the negative class (dominated by the background).

3. If a pedestrian stays in a particular image location for a long time, all the pedestrian patches in this duration will be incorporated as "non-pedestrian" data and the resulting classifier at that image location would then learn to classify pedestrians as "non-pedestrians" with high probability (thereby decreasing the recall of the system).

4. Even though training one classifier per image location simplifies the task of each classifier, the combined complexity of all the classifiers is still much higher. And due to the fact that negative data are not *shared* among the individual classifiers, it can result in overfitting at the system level (even if there is no overfitting at the individual classifier level).

Because of these problems, the system may result in low recall and low precision simultaneously, especially in complex surveillance-type scenes with multiple object categories. Coincidentally, they evaluate their method only on relatively simple indoor scenes where there is only one class of moving objects as shown in Figure 3.9. In fact, rather than "pedestrian detection", the system is more similar to the traditional background subtraction and if applied to more complex scenes, it would not be much different than block-based background subtraction approaches such as [49]. A similar system is also proposed by Roth *et al*. [91].

Stalder *et al*. [102] extend [46] by updating both the positive and negative classes in each image location (*i.e*. for each classifier) and proposing different update strategies than [46]. The positive class for each image location is updated using the current patch if it is verified by a fixed generic detector (which is a global detector independent from the grid classifiers) or 3D context (*e.g*. assumption of a common ground plane). Negative class for each image location is updated by background images at that location obtained by a long-term generative (pixel-based) background subtraction algorithm. Although the paper proposes more complex update heuristics than [46] for the classifiers, it also somewhat defeats the original purpose of having these grid-based classifiers, which is to make the task of each classifier simple and robust to drifting (at least for the positive class) by adopting fixed updating strategies. Compared to [46], their approach opens up the possibility of positive class drifting. Moreover, updating the negative class with the results of background subtraction introduces errors associated with most pixel-based and generative background subtraction methods. This problem is minimised in [46] by avoiding pixel-wise background modelling and instead, by modelling the large neighbourhood of pixels in a discriminative fashion. Therefore, in [102], the need for grid-based classifiers is no longer obvious. Furthermore, it also still shares a few limitations of [46]. And lastly, it requires the assumption and estimation of a single ground plane and 3D context which may not be readily available.

3.2.3.2 Semi-supervised learning for object detection

In this section, we briefly review work on semi-supervised learning of object detectors for videos. However, as mentioned in Section 2.6.1, this work solves a different problem than domain adaptation (*i.e*. semi-supervised learning setting assumes that there is no source domain and some labelled data are always given in the target domain) but we include these here for completeness.

Levin *et al*. [65] propose a co-training [9] approach for semi-supervised learning of vehicle detectors in video. Given some labelled data in the target scene, firstly, a pair of

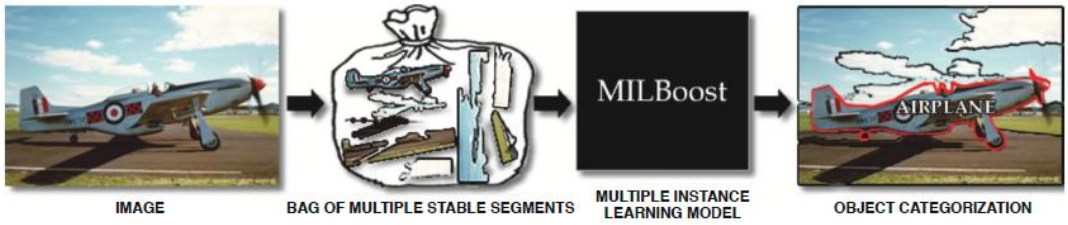


Figure 3.10: An image is represented by a *bag* of multiple stable segments which is obtained by collecting the outputs of different segmentation algorithms and various segmentation parameters with the assumption that one of the segmentations in the bag would correctly correspond to the aeroplane. Then by looking at multiple such bags corresponding to training images where each image contains an aeroplane, the aeroplane category can be inferred and segmented in each of the images. Figure taken from [38].

car detectors is trained; one of the pairs is trained on data for whose feature extraction is performed on original images and for the other, background subtracted images instead of the original images are used. Then these two classifiers are used to teach and improve each other by "feeding" one the confident detections of the other and retraining the classifiers. They tested their methods on videos of vehicles on a highway captured by a surveillance camera. A similar co-training system is presented by Javed *et al*. [55] by using online boosting.

Rosenberg *et al*. [89] use iterative self-training for semi-supervised learning of an eye detector. For the "oracle" (*i.e*. for selecting which examples to include for each iteration of self-training), instead of using the detector's own confidence, the system uses nearest neighbour scores of all examples in the current dataset to the detection, in an attempt to make the oracle independent from the detection. However, the oracle is still not independent because it is derived from the same dataset that the detector was trained from. Their approach can also be seen a type of co-training where two classifiers have two different classifier types (*i.e*. inductive biases) and one of the classifiers is fixed.

Ali *et al*. [2] propose an iterative self-training algorithm based on Adaboost for semi-supervised learning of a pedestrian detector in a video, given sparsely annotated video (*i.e*. a small subset of all the frames in the video are labelled). Examples to include for each iteration of the self-training is determined by track smoothness. The method is however limited to Adaboost and not applicable to other types of classifiers.

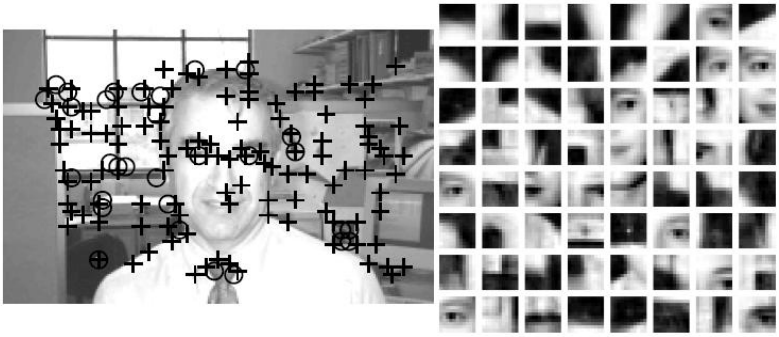


Figure 3.11: On the left is an example of interest point detection on an image containing a face. The right picture shows a set of distinctive parts discovered by clustering the patches corresponding to interest points across multiple training images containing faces. Figure taken from [114].

3.3 Weakly Supervised Learning For Object Detection

Galleguillos *et al*. [38] propose a weakly supervised approach to learn object detectors given weakly labelled images. In their case, weakly labelled images are considered as images containing the desired objects but the exact locations and spatial extent of those objects are not specified. However, the method does require the objects to be spatially occupying the major portion of the images for their algorithm to work well. To our knowledge, they are the first to use the idea of "multiple stable segmentations" and Multiple Instance Learning (MIL) for the purpose of training object detectors using weakly labelled images. Multiple stable segmentations is an idea that in any image containing an object of interest, an ensemble or bag of segmentations obtained by multiple segmentation algorithms and different segmentations parameters will most likely result in the object being *correctly* segmented in at least one of these segmentations in the ensemble. Each image containing an object can therefore be associated with a bag of segmentations from which one of them corresponding to the desired object. This is a much better and useful prior information than not having any information about the object in the image. Since an image containing an object can be represented with a bag (of possible objects), MIL can be used to learn the most consistent object category by optimizing across multiple such images and corresponding bags. This is illustrated in Figure 3.10.

Weber *et al*. [114] propose another weakly supervised training approach to learn human face models and models of rear views of cars. Again, similar to [38], their method assumes that each object occupies the major portion of the corresponding training image. They represent an object as a constellation of parts where parts are detected by an interest point detector. Distinctive parts are discovered by clustering the detected parts (repre-

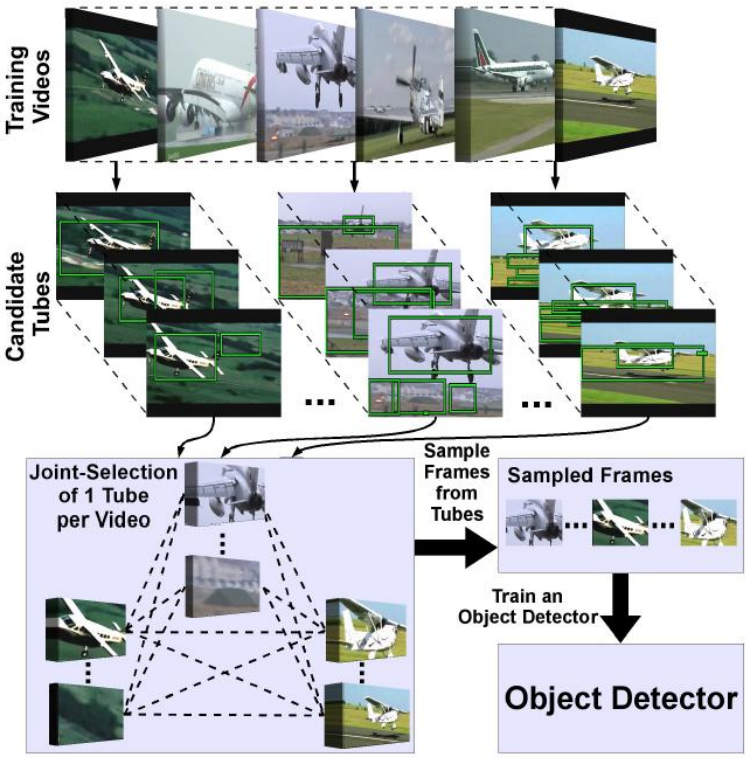


Figure 3.12: Overview of the weakly supervised learning approach by Prest *et al*. [84]. Figure taken from [84].

sented by features extracted from regions centred at the interest points) in the training images. This is shown in Figure 3.11. Then object classes are learnt by searching for parts and geometry of parts that are consistent across the training images. The method requires high resolution images (for reliable part detection) and is dependent upon the interest point detector to consistently and correctly fire on actual part-like locations in images. Moreover, if the background clutter is high, the system may not correctly learn the desired models.

Blaschko *et al*. [7] and Pandey and Lazebnik [81] propose latent-SVM-based weakly supervised training algorithms where the bounding boxes of objects are treated as latent variables to be inferred during training. The general problem, however, with these approaches is that the resulting optimization function is very prone to get stuck in bad local optima unless there is a good initialisation. Although they propose some heuristics for initialising such models, they are usually application and object-specific. Moreover, the training algorithm can also be very computationally expensive.

Prest *et al*. [84] propose a weakly supervised learning of object detectors from short YouTube video clips where each video clip is assumed to contain an object of interest

moving about. A diagram illustrating the overview of their approach is shown in Figure 3.12. Their method first identifies candidate spatio-temporal tubes from which at least one of them is very likely to contain a moving object of interest in each video clip. Then from many sets of these candidate spatio-temporal cubes from multiple video clips, a consistent class of spatio-temporal cubes is found by jointly considering all the spatio-temporal candidate cubes across all the training video clips and minimising an objective function. Similar to many other weakly learning approaches, their approach also has an implicit assumption that objects of interest occupy the majority of the spatio-temporal volume in the video clips and the optimization algorithm can get stuck in bad local optima without a suitable initialisation which is non-trivial.

3.4 Relation of this Thesis to Prior Research

In this section, we put into context the contributions made in this thesis and directly compare them to the relevant papers that have been discussed in detail in Section 3.2 and Section 3.3. Furthermore, where appropriate, we also identify the most relevant prior work that we would quantitatively compare our proposed algorithms with, in the experimental results sections of Chapters 4, 5 and 6.

3.4.1 First Thesis Contribution (Chapter 4)

Although we are dealing with object detection in videos, for this contribution, we refrain from using spatio-temporal continuity and cues available from video (such as background subtraction and tracking) and deal only with the *manifold assumption* of data. We do this because it is of interest to know how well the manifold assumption alone can perform. However, it is likely that better performance could be obtained by exploiting information from those spatio-temporal cues (which is explored in the second contribution). To our knowledge, our contribution is the first work in using manifold learning (by means of deep feature learning) for domain adaptation of object detectors in the context of videos. Therefore, we compare our proposed algorithm with state-of-the-art approaches that make use of a similar manifold assumption and metric learning, and not with iterative self-training algorithms (which will be compared with the second thesis contribution in Chapter 5).

The approaches proposed by Saenko *et al*. [94] and Kulis *et al*. [60] are not closely related to our approach because their methods are about *supervised* domain adaptation and in addition to that, their method requires mapping of pairs of examples from source and target domains. In contrast, our approach is *unsupervised* domain adaptation and

requires no such mapping. The system proposed by Mirrashed and Rastergari [74] is also not directly relevant because they do not make use of a manifold assumption and learning.

One of the most relevant works to our research in this area is by Gopalan *et al.* [45] who 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.* [44] improves on [45] by giving a kernel version of [45]. We therefore quantitatively compare our approach proposed in Chapter 4 with [44] (and therefore also indirectly with [45]). However both [44, 45] are dealing only with domain adaptation for image classification as opposed to our algorithm which works with domain adaptation for object detection. Moreover, unlike our algorithm, their approaches do not learn deep representations required for manifolds that are highly non-linear.

3.4.2 Second Thesis Contribution (Chapter 5)

As can be observed in Section 3.2.2, the overwhelming majority of the state-of-the-art research for domain adaptation of object detectors in videos use self-training in one form or another [10,54,58,59,73,99–101,110,111,113,115]. In order to adapt a generic pedestrian detector to a specific scene, a typical system would run the generic detector on some frames in a video, then score each detection using some heuristics and afterwards, add the most confident positive and negative detections to the original dataset for retraining. This process is repeated over multiple iterations. Each of these approaches suffers from a subset of the following problems:

1. The need to manually determine and set thresholds for "the most confident" detections, "the least confident" detections, low precision and high recall settings and so on.

2. Many of them only work with specific types of classifier (such as Adaboost, cascaded classifiers and Multiple Instance Learning).

3. Most of them need setting of the number of iterations for the iterative self-training.

4. Many of them are prone to drifting since wrongly labelled examples in one iteration could make the detector become progressively worse in the following iterations.

5. Most of them require the presence of the original dataset for retraining. This is expensive especially for large datasets and many a time, we may only have a generic

detector (which can be a classifier of any type) but not the generic *dataset* itself (due to copyright reasons, *etc*.).

6. Most approaches have several sensitive parameters to set and tune. And these parameters change for different videos (or scenes), many of them cannot be set automatically (for unsupervised domain adaptation) and for some, it is non-trivial to tune them automatically without extensive and very expensive cross validation.

7. Many of the approaches do not work well with low-resolution far-field surveillance videos which we tackle in this thesis.

8. Some of them require labelled data (*i.e*. supervision) in the target domain, *i.e*. they are supervised domain adaptation approaches.

Our second contribution in the thesis does not suffer from the problems listed above. We now go through each of the related works discussed Section 3.2.2 and briefly relate it to our proposed algorithm.

Bose and Grimson [10] evaluate their approach on far-field surveillance scenes which is similar to the type of problem that is addressed in this thesis. However, most of the improvement of their detector adaptation comes from using a different and better feature extraction *at* test time. In contrast, in our algorithm, the majority of the benefit of domain adaptation is derived from the systematic and effective collection of scene-specific positive and negative examples. Therefore, unlike [10], our approach can still improve performance further by extracting new and better features specific to the scene after collecting the scene-specific positive and negative data. The approach by Wu and Nevatia [115] only works with part-based detectors, so it is not really suitable for the majority of holistic object detectors that we focus on in this thesis. The Multiple Kernel Learning approach of Kemhavi *et al*. [58] is expensive at test time as opposed to our algorithm which can generate a linear classifier (or any type of classifier) for test time. Wang *et al*. [113] and Sharma *et al*. [99] deal with domain adaptation for pedestrian detection, however their approach is not likely to work well for the low resolution videos tackled in this thesis. The *supervised* domain adaptation approach of Jain and Farfade [54] using cascade classifiers solves a different problem than ours which is about *unsupervised* domain adaptation. The paper by Mirrashed *et al*. [73] is also not very relevant to us because it requires multiple (*i.e*. at least two) source domains whereas in our thesis, we assume that only one source domain is available. The approach of Shu *et al*. [101] may work poorly for videos with small pedestrians.

The two most relevant state-of-the-art methods to our contribution are the approaches by Wang and Wang [111] and Wang *et al*. [110]. They deal with unsupervised domain adaptation of pedestrian detectors trained on a generic image dataset to far-field videos where pedestrians are small or medium-sized. Moreover, they provide two long far-field video datasets for quantitative evaluation for domain adaptation of pedestrians detectors. Therefore, we use these datasets in this thesis and also compare our algorithms with theirs. Apart from quantitatively comparing our algorithm with [110,111], we also compare with a variation of the approach of Nair and Clark [75]. Even though their method cannot be considered as a domain adaptation approach, a modification of their approach provides a convenient and useful baseline for domain adaptation using "naive" background subtraction.

3.4.3 Third Thesis Contribution (Chapter 6)

Compared to training object detectors using strong supervision, the literature concerning weakly supervised training is limited. Furthermore, most of the literature on weakly supervised learning in images solves a different problem than our proposed approach. In the existing approaches, supervision is given in the form of image-level labels where the exact location and spatial extent of objects of interest are considered unknown and treated as latent variables to be inferred from data during training. One of the ways of solving this problem is by formulating it as Multiple Instance Learning (MIL) [4, 23] in which supervision labels are given at the *bag* level rather than at the instance level. Each positive bag is assumed to contain at least one positive instance and each negative bag is assumed to contain all negative instances. In order to generate positive bags and because the space of all possible object locations and sizes is too large to be tractable during training, many existing approaches use an ensemble of low-level segmentations to generate numerous candidate regions with the assumption that at least one of them contains the desired object [16, 38]. The output of such a system, however, depends heavily on the results of segmentation. Our algorithm does not suffer from this problem.

Furthermore, most existing approaches work with datasets where an object occupies a large central portion of each image in most of the training images [16, 18, 38, 77]. This is in contrast to our approach which is dealing with far-field videos where there are often multiple objects of varying sizes in each frame and each object occupies only a tiny portion of a frame. Moreover, our approach can work with low-resolution objects that do not allow sophisticated part-based modelling and discovery.

Deselaers *et al*. [22] propose an iterative algorithm to learn object classes from weakly

supervised images using a conditional random field that progressively adapts to the new classes. Chum and Zisserman [18] give an algorithm that locates image regions corresponding to object classes of a set of training images by optimizing an objective function that computes similarity between pairs of images. Considering classifier parameters and subwindows of objects jointly as latent variables in an SVM classification objective function, Nguyen *et al*. [77] optimize the function to infer the variables. Weakly supervised learning is tackled as a structured output learning framework in [7]. All of the aforementioned approaches deal only with images and do not make use of information that can be exploited in surveillance-type videos.

Recently, Prest *et al*. [84] propose a weakly supervised learning approach for YouTube video clips. Their approach, which is essentially an extension of [22] to video, solves a fundamentally different problem from our contribution in that they assume that small independent video clips are the training data and each video clip contains the desired object class in a large proportion of the spatio-temporal volume, whereas we do not have any such assumption and setting, and we are dealing with long videos captured by a static uncalibrated camera overlooking a scene.

Chapter 4

Unsupervised Detector Adaptation by Joint Dataset Feature Learning

4.1 Overview

In this chapter, we present the first major contribution in the thesis; a novel unsupervised domain adaptation algorithm is proposed to automatically adapt a pedestrian detector trained on a generic image dataset to a video using *joint dataset deep feature learning*. This is achieved by using state-of-the-art deep learning to learn the nonlinear manifold jointly spanned by the source image dataset and the sampled data of the target video. The intuition is that by learning a representation of this manifold and training a classifier on data in this representation, the resulting detector would generalise well for the target scene. This can then be used as a scene-specific detector. Our algorithm does not require any background subtraction or tracking in the video and relies purely on the "power" of manifold learning for domain adaptation. Experiments on two challenging video datasets show that our algorithm is effective and outperforms the state-of-the-art approach.

4.2 Contributions

For this chapter, we can summarise the novel contributions as follows:

1. An algorithm that adapts a pedestrian detector from a labelled generic image dataset to an unlabelled video using only the manifold assumption.

2. An application of state-of-the-art unsupervised deep feature learning for the task of domain adaptation of a pedestrian detector to 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).

3. A simple biased sampling technique to minimise the problem of large class imbalance between samples from the pedestrian class and those from the non-pedestrian class in video. Without the proposed sampling algorithm, (naive) random sampling of data in videos will result in almost all samples to be from the non-pedestrian class.

4. An effective technique to automatically set the structure of the deep network without hand tuning.

5. The integration of all of the above components into a system.

4.3 Proposed Approach

4.3.1 Overview

The overview of the algorithm is illustrated in Figure 4.1. We are given a generic pedestrian image dataset as the source dataset and a target video. The goal is to obtain a scene-specific pedestrian detector that will perform better on the target video than a pedestrian detector trained on the generic dataset. The algorithm is made up of two stages:

1. Unsupervised deep feature learning. The manifold spanned by the combination of the source and the target dataset is learnt using deep learning. During this stage, labels from the source dataset are not used.

2. Non-linear projection and classifier training. In this stage, only the source dataset (and the labels) is used.

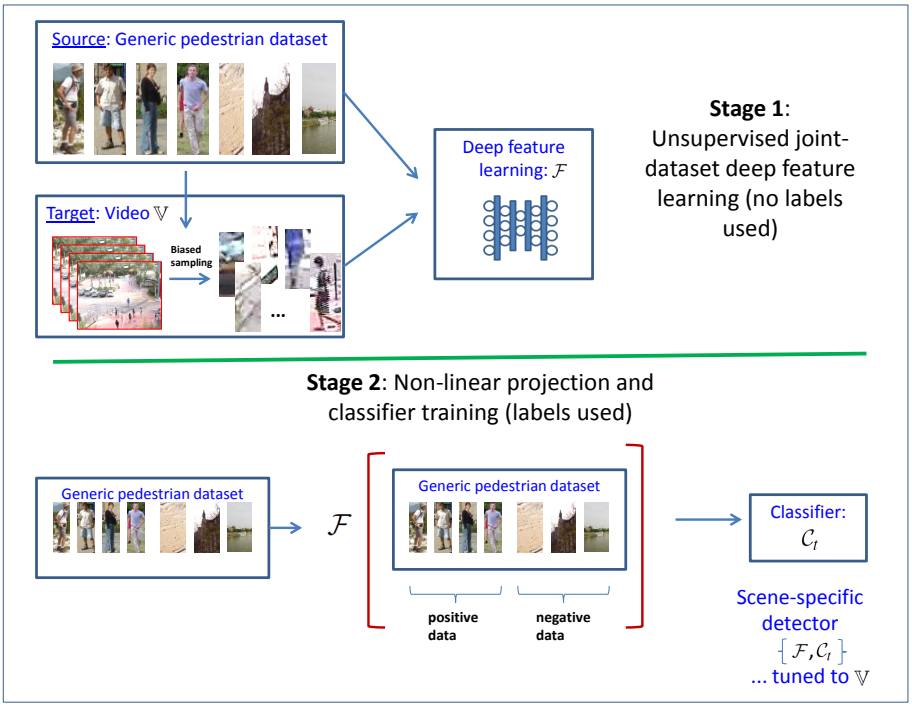


Figure 4.1: Overview of the proposed algorithm. The inputs to the algorithm are a source dataset (labels available) and a target video $\mathbb{V}$ (no labels) which we want the detector to adapt to. The output of the algorithm is a scene-specific detector (SSD) which is tuned to $\mathbb{V}$. The SSD is made up of a function $\mathcal{F}$ which projects a given feature vector onto the learnt non-linear manifold and a classifier $\mathcal{C}_t$. Note that all extracted patches from the generic dataset or the target video, shown in the figure, are actually represented as feature vectors by applying a feature extraction function $\mathcal{H}$. The function $\mathcal{H}$ is not explicitly shown in the figure for clarity.

Firstly, let $\mathcal{H}$ be a function for feature extraction on $N_r \times N_c \times 3$ colour image patches, *i.e.* $\mathcal{H} : \mathbb{R}^{N_r \times N_c \times 3} \rightarrow \mathbb{R}^{N_{\text{base}}}$, where N_r and N_c are the number of rows and columns of a colour image patch respectively and N_{base} is the number of dimensions of the feature vector. Let the generic labelled pedestrian dataset (assuming features extracted) be

$$\mathbf{X}_s^l = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$$

where $\mathbf{x}_i \in \mathbb{R}^{N_{\text{base}}}$. The dataset $\mathbf{X}_s^l$ is associated with a set of labels

$$\mathbf{Y}_s = \{y_1, y_2, \dots\}$$

where $y_i \in \{1, 0\}$. Let the target video $\mathbb{V} = [I_1, I_2, \dots]$ be a sequence of frames.

Algorithm 4.1 describes the detector adaptation process. For learning a manifold that is spanned by the source and the target datasets, firstly a joint unlabelled dataset, $\mathbf{X}_{s+t}^u$,

is required. Combining the source and the target dataset is not trivial because the target dataset is a video that typically contains a very large number of possible unlabelled patches that could be extracted, from which only a very small percentage correspond to pedestrians.

Therefore, unlabelled target data, $\mathbf{X}_t^u$, are sampled from $\mathbb{V}$ using the biased sampling technique detailed in Algorithm 4.2. Then $\mathbf{X}_{s+t}^u$ is obtained by combining the source dataset $\mathbf{X}_s^l$ with the sampled data $\mathbf{X}_t^u$. Labels from $\mathbf{X}_s^l$ are ignored at this stage. $\mathbf{X}_{s+t}^u$ serves as input to the manifold learning algorithm (using deep feature learning) described in Algorithm 4.3.

The manifold learning algorithm produces, as output, a function $\mathcal{F} : \mathbb{R}^{N_{\text{base}}} \rightarrow \mathbb{R}^{N_{\text{deep}}}$ which takes in a base feature vector and produces a feature vector of much smaller dimension by projecting the base feature vector onto the learnt manifold. We then project $\mathbf{X}_s^l$ into this space and together with $\mathbf{Y}_s$, train a classifier, $\mathcal{C}_t : \mathbb{R}^{N_{\text{deep}}} \rightarrow \{1,0\}$, on these projected data. The algorithm can work with any type of base features and classifier combination. For simplicity, we use HOGs and a linear Support Vector Machine (SVM) respectively.

Algorithm 4.1 Detector adaptation overview

Input: $\mathbf{X}_s^l$, $\mathbf{Y}_s$, $\mathbb{V}$, $\mathcal{H}$
Output: Scene-specific detector $= \{\mathcal{F}, \mathcal{C}_t\}$
1: $\mathbf{X}_t^u \leftarrow \text{BiasedSamplingDataFromVideo}(\mathbb{V}, \mathbf{X}_s^l, \mathbf{Y}_s, \mathcal{H})$
2: $\mathbf{X}_{s+t}^u \leftarrow \mathbf{X}_s^l \cup \mathbf{X}_t^u$
3: $\mathcal{F} = \text{LearnManifold}(\mathbf{X}_{s+t}^u)$
4: $\mathcal{C}_t = \text{LearnClassifier}(\mathcal{F}(\mathbf{X}_s^l), \mathbf{Y}_s)$
5: **return** $\{\mathcal{F}, \mathcal{C}_t\}$

4.3.2 Sampling Representative Unlabelled Data from Target Video

Algorithm 4.2 Biased sampling of unlabelled data from video

Input: $\mathbb{V}$, $\mathbf{X}_s^l$, $\mathbf{Y}_s$, $\mathcal{H}$
Output: Sampled data, $\mathbf{X}_t^u$
1: $\mathcal{C}_s = \text{LearnSVM}(\mathbf{X}_s^l, \mathbf{Y}_s)$
2: Random sample k frames from $\mathbb{V}$
3: Run sliding-window detector with $\mathcal{C}_s$ on the k sampled frames
4: $\mathbb{P} \leftarrow$ Random sample patches from the positive detections of detector
5: $\mathbf{X}_t^u \leftarrow \mathcal{H}(\mathbb{P})$
6: **return** $\mathbf{X}_t^u$

For state-of-the-art domain adaptation approaches for *image classification* that involve learning a common feature representation, sampling data from the target dataset is

straightforward and just random sampling would suffice. However, since we are dealing with a far-field video as the target dataset (and object detection), naive random sampling of unlabelled patches $\mathbf{X}_t^u$ from $\mathbb{V}$ in the space of multi-scale sliding windows is not a good idea because the probability of obtaining pedestrian patches, *i.e.*

$$P(\text{label=pedestrian}|\text{random patch})$$

is extremely low and is given by:

$$\frac{\text{Average number of pedestrians in a frame}}{\text{Number of multi-scale sliding windows in a frame}} \tag{4.1}$$

Assuming that the average number of pedestrians is each frame is 10 and the number of multi-scale sliding windows in each frame is $500,000$, then the probability of randomly sampling a correctly localised pedestrian is less than 0.00005.

Although the feature learning stage requires only unlabelled data, the samples should still be representative of both non-pedestrian and pedestrian patches in $\mathbb{V}$. Note that we are *not* concerned with any supervision labels during this sampling stage; the objective is simply to get an overall unknown mixture of pedestrian and non-pedestrian patches from $\mathbb{V}$ without needing pedestrian/non-pedestrian labels for each sampled data.

Therefore, we use a biased sampling strategy as given in Algorithm 4.2. The patches sampled with the algorithm would then be a mixture of pedestrians and non-pedestrians (corresponding to false positives of the detector).

4.3.3 Learning the Manifold of the Joint Unlabelled Dataset

Algorithm 4.3 Manifold learning using deep feature learning

Input: Unlabelled joint dataset (base features), $\mathbf{X}_{s+t}^u$
Output: Learnt non-linear projection function, $\mathcal{F}$

1: Learn and apply PCA on $\mathbf{X}_{s+t}^u$ and keep 99% variance
2: $N_{\text{deep}} \leftarrow$ estimate intrinsic dimension of $\mathbf{X}_{s+t}^u$ using [66]
3: $\mathcal{A} \leftarrow \text{SetUpAutoEncoder}(N_{\text{base}}, N_{\text{deep}})$
4: $\mathcal{A} \leftarrow$ Initialise $\mathcal{A}$ using [43]
5: $\mathcal{A} \leftarrow$ Minimise $\text{LossFunc}(\mathcal{A}, \mathbf{X}_{s+t}^u)$ using mini-batch L-BFGS
6: $\mathcal{F} \leftarrow$ Remove the decoder part of $\mathcal{A}$
7: **return** $\mathcal{F}$

After obtaining the joint unlabelled dataset $\mathbf{X}_{s+t}^u$ as described previously, we now learn a common feature representation across the source and the target dataset, *i.e.* $\mathbf{X}_{s+t}^u$. Unlike most state-of-the-art techniques (described in Section 3.2.1) that propose various

objective functions for learning common feature representations specialised for domain adaptation, in our approach, a common feature representation is obtained by explicitly making the manifold assumption and learning the manifold using a much more general technique, namely, deep (feature) learning.

Deep learning is a subset of machine learning techniques utilised for learning hierarchies of features (*i.e*. representations) to obtain high-level abstractions of data. Although deep learning is very general with many emerging applications, we find that deep learning, applied as part of our proposed algorithm, outperforms state-of-the-art domain adaptation techniques that learn common feature representations that are specially designed by researchers for use with domain adaptation.

We now briefly describe manifold learning. Although visual data (*i.e*. observations) are often represented as points in high-dimensional vector space, they exist in a much lower-dimensional *manifold* and can therefore be represented much more compactly. This lower-dimensional manifold is a direct consequence of the inherent *structure* that characterise and that exists in the natural world, resulting in strong (typically non-linear) correlations between data dimensions. *Manifold assumption* is the assumption that real-world data lie approximately on a smooth low-dimensional manifold embedded into a high dimensional space. We can use this assumption and explicitly learn the manifold which allows recovery of the underlying structure parameters that generated the given high-dimensional observations.

By learning the underlying manifold and training a classifier on the labelled source data $(\mathbf{X}_s^l, \mathbf{Y}_s)$ in this space, we hope to build a classifier that would perform well on the target video.

In order to learn the manifold, we use a deep autoencoder [6] which is one of the most popular deep learning methods. A deep autoencoder is a *network* of nested compositions (*i.e*. layers) of non-linear functions. It takes an input data vector, *encodes* it by recursively projecting it through some non-linear functions and then *decodes* by recursively projecting the encoded data through the rest of the non-linear functions to reconstruct the input data.

The network is designed in such a way that the dimension of the encoded data decreases with each encoding layer until a *bottleneck* layer is reached which has the smallest number of dimensions. Then the decoding layers increase in the number of dimensions gradually. Due to the presence of the bottleneck layer, the autoencoder is forced to learn a compact representation of the input data which can only be achieved by some sort of generalisation on the input data.

The input data projected through such nested compositions of non-linear functions

give "deep features" which can be considered as high-level abstract feature representations of the input data. Non-linear classifiers such as non-linear Support Vector Machines (SVMs), Random Forests, Adaboost and 1-hidden-layer Multilayer Perceptrons also implicitly learn non-linear functions in the internal representations of the classifiers. However,

1. They require all training data to have supervision labels. This means that we cannot use them for our purpose since we do *not* have labels for the target video $\mathbb{V}$, *i.e*. we are working with $\mathbf{X}^{u}_{s+t}$.

2. The non-linear functions that they learn are *not* deep. In terms of representational power, they are only equivalent to 1-hidden-layer neural networks or less. For example, for SVMs, the non-linearity learnt is not even as flexible as a 1-hidden-layer neural network because the non-linear functions for an SVM are fixed to be data from the training set and only the coefficients to combine the functions are learnt.

One way to avoid these two limitations is by using a deep autoencoder, which is an unsupervised approach. A deep autoencoder is a generalisation of many dimensionality-reduction and sparse-learning methods in machine learning. For instance, PCA can be considered as a type of autoencoder with one hidden layer, linear activation functions and Mean Squared Error (MSE) loss function.

Since a deep autoencoder allows learning of deep non-linear features of data without requiring any labels during training, large amounts of data can be used, providing sufficient data for estimation of a large number of unknown parameters. This, combined with recent advances in better initialising deep neural networks [30, 43, 70, 76], has made deep learning effective and practical for various computer vision applications.

Despite the large amounts of unlabelled data available for training, deep networks are still prone to poor[1] local optima. One of the recent breakthroughs in training such deep networks is by using a better weight initialisation scheme. This is accomplished by layer-wise stacked pre-training where each layer of the deep network is greedily trained by using the output of the previous layer as input for the current layer [30]. Then, these weights are used as the initialisation for the joint training of all the layers.

However, even more recent research [70, 76] has questioned the need for such a layer-wise greedy pre-training, arguing that one of the main problems with deep networks is "pathological curvature" which looks like local optima to first-order optimization techniques such as gradient descent, but not to second-order optimization methods. Another

[1] In deep learning, the goal is not to find the global optimum but just a "good enough" local optimum [30].

advantage of second-order optimization methods is that they do not require setting or tuning of the learning rate. However, despite these obvious advantages, they are not widely used because they share one common problem: they are not suitable for moderate to large amounts of data because they require computing and storing of the Hessian matrix during the optimization.

We overcome the aforementioned problems in a novel combination of the following techniques:

1. The optimization is done by the limited-memory version of Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [68] which is an approximated second-order method. Due to it being a second order optimization algorithm, the need for layer-wise pretraining is eliminated, the probability of getting stuck in poor local optima is minimised and there is no need to tune the learning rate.

2. We use a *mini-batch* training technique. Mini-batch training is the compromise between (full) batch training and online training. In each iteration, rather than presenting the optimization algorithm with the entire training data (as in batch training) or with a single training data point (as in online training), we present it with a random portion of the training data. This has two advantages:

   (a) The data fits into the memory and the optimization is fast.

   (b) It has been shown in recent literature (*e.g.* [11, 96]) that the *stochastic noise* imparted by mini-batch training could help avoid some sensitive local optima by essentially "blurring" out those local optima.

3. We adopt the weight initialization proposed in [43]. This further increases the probability of obtaining a better local optimum.

Although the idea of using L-BFGS or training using mini-batch is not novel on its own, to the best of our knowledge, bringing together and integrating these state-of-the-art findings and the combination of the three methods outlined above (in addition to the more significant fact that we are simply using deep learning as part of our domain adaptation algorithm) is novel.

Manifold learning using deep learning is formalised in Algorithm 4.3 and is explained in more detail below.

4.3.3.1 Data normalisation

As is common in the feature learning literature, we first perform Principal Component Analysis (PCA) and keep 99% of the total variance (which retains almost all of the in-

formation) to normalise and condition the data for faster convergence during subsequent optimization. All data is projected onto the Principal Component space before being fed to the autoencoder. We however omit the PCA projection step in all the figures and descriptions for the sake of clarity.

4.3.3.2 Setting up the deep autoencoder architecture

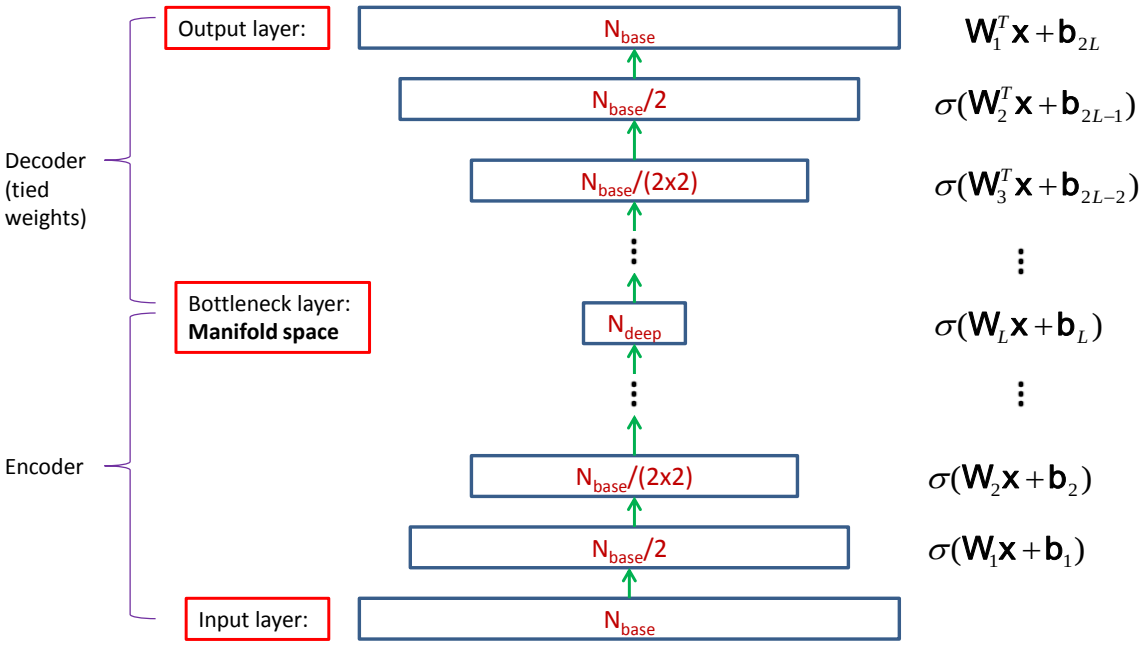


Figure 4.2: The deep autoencoder architecture. See text in Section 4.3.3.2 for the explanation of the notation and the architecture.

The first step for deep learning is to set up the deep autoencoder architecture and this involves setting the number of layers, the size of each layer and the activation function in each neuron of each layer. The architecture is one of the most crucial aspects of deep learning because it determines the space and the complexity of the family of functions that are to be learnt.

If the layer sizes (*i.e.* the number of neurons in the layers which determine the breadth of the network) are too small, then the space of functions will be limited and there will be too many poor local optima resulting in limited generalisation power [61]. The number of layers determines the depth of the autoencoder and generally, increased depth allows learning of non-linear functions in a more efficient way (since deep compositions of non-linear functions results in higher non-linearity with fewer parameters). However, too many layers can give problems in the optimization due to issues with the gradient signal getting lost in the layers in the back-propagation algorithm (commonly called the "van-

ishing gradient problem" [50]), although the L-BFGS and mini-batch training which we use can mitigate against this.

There is a commonly used heuristic in the deep learning literature for setting up a deep autoencoder and that is to decrease the layer sizes from the input layer (which is equal to the number of dimensions of the input data) to the bottleneck layer. These are the encoding layers. Then the layer sizes are mirrored (and therefore made symmetric) and they increase until the output layer is reached (whose size is equal to the size of the input layer). These are the decoding layers. Another widely used heuristic is to tie the weights (but not the biases) of the encoding layers to the weights of corresponding (symmetric) decoding layers. This is reasonable (since decoding can be interpreted as the exact inverse of encoding) and greatly reduces the number of parameters to be learnt during optimization. We adopt both of these heuristics in our work.

However, setting the exact sizes of the individual layers (including the size of the bottleneck layer) and the number of layers is still arbitrary in the literature. Here, we propose a more principled algorithm:

1. Firstly, we estimate the intrinsic dimensionality of the data $\mathbf{X}^u_{s+t}$ using [66]. Let this be N_{deep}.

2. For encoding layers, starting with the size of the input layer (which is known), each hidden layer has half of the number of hidden neurons as its previous layer and this is repeated until a layer of size equal to or less than N_{deep} is achieved. If less than N_{deep}, the layer size is set to N_{deep}. This is the size of the bottleneck layer. Now we have set the layer sizes of all the encoding layers.

3. For decoding layers, the layer sizes mirror those of the encoding layers.

This is shown Figure 4.2. The network structure is obtained automatically and systematically and we do not manually tune it. There are a total of L hidden nonlinear layers in the encoder (which produces a total of $2L$ layers feed-forward deep network).

The last design decision to be made as part of the architecture setting is the neuron *activation function* (which is also known as the *transfer function*). We set all the neurons in all the layers to have the same activation function and the activation function is made to be non-linear because we want the autoencoder to learn (nested) non-linear functions. The non-linear activation function used is the hyperbolic tangent function (represented in Figure 4.2 with the function σ). This is one of the most common activation functions for neural networks and has been shown to work well in the literature (especially for deep autoencoders). The function is given in Equation 4.2 and shown in Figure 4.3.

$$\begin{aligned}\sigma(z) &= \frac{\sinh z}{\cosh z} \\ &= \frac{e^{(z)} - e^{(-z)}}{e^{(z)} + e^{(-z)}} \\ &= \frac{e^{(2z)} - 1}{e^{(2z)} + 1}\end{aligned} \tag{4.2}$$

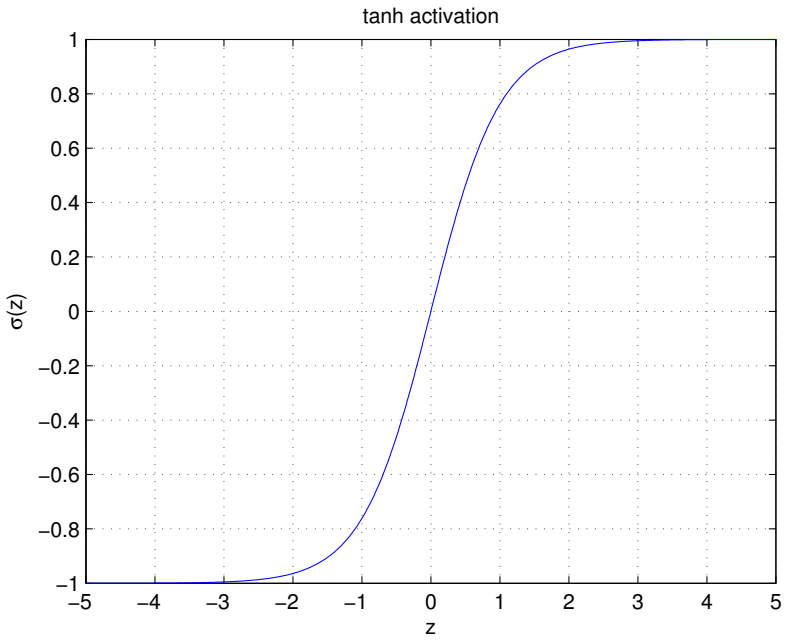


Figure 4.3: Hyperbolic tangent activation function to impart non-linearity in the deep network. The input signal z is "squashed" (*i.e.* non-linearized) such that the output $\sigma(z)$ is within the range $[-1,1]$. Note that the function acts on each dimension of the input vector independently.

4.3.3.3 Initialising the deep autoencoder architecture

After the network has been set up, we randomly initialise it using the method of Glorot and Bengio [43], which has been shown to perform well for deep networks. Biases are initialised to zero and weights $\mathbf{W}$ for a layer are initialised by uniformly sampling as follows:

$$\mathbf{W} \sim \mathcal{U}\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{4.3}$$

where n_j is the size of current network layer and n_{j+1} is the size of next layer. Equation 4.3 is obtained in [43] by studying and analysing the saturation of hidden neurons and by observing the flow of gradient signals across network layers as the activation functions and network initialisation are varied.

4.3.3.4 Network optimization

In order to train the autoencoder, the network in Figure 4.2 can be mathematically written as a smooth differentiable multivariate loss function which should be minimised.

This is given in Equation 4.5; m refers to the number of data points in each mini-batch, $\mathbf{W}_j$ is the affine projection matrix for layer j, $\mathbf{x}_i$ is a column vector of data point i, and $\mathbf{b}_j$ is a vector of biases for layer j. The function $\mathcal{A}$ is defined in Equation 4.4; it projects a given input data point $\mathbf{x}$ to any layer j in the deep network with the given network weights $\mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, \mathbf{b}_{2L}$. $\mathcal{A}$ is a function that will recursively call itself and project $\mathbf{x}$ from the input layer up to layer j.

During each iteration of the optimization algorithm, given the training data, the network and the current weights (being optimized), the following information is required:

1. The loss function value. This is obtained analytically.
2. The gradient vector obtained by the first-order derivative of the loss function. This can also be obtained analytically.
3. The second-order derivative of the loss function. This is a matrix approximated by L-BFGS.

The loss function involves first encoding the data by feed-forward projection of data through L encoding layers until the bottleneck layer is reached. After that, the decoding stage attempts to reconstruct the original data. Then the Mean Squared Error gives the loss function value. To obtain the loss function gradient vector during optimization, we need to find the (first-order) derivative of the loss function. Since the loss function consists of deeply nested composites of non-linear functions, the chain-rule is used. This can be efficiently obtained by the back-propagation algorithm [93] which is a type of dynamic programming technique.

$$\mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x},j)$$
$$= \begin{cases} \sigma(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1) & j=1 \\ \sigma(\mathbf{W}_j\mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x},j-1)+\mathbf{b}_j) & 1<j\leq L \\ \sigma(\mathbf{W}_{2L-j+1}^T\mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x},j-1)+\mathbf{b}_j) & L<j<2L \\ \mathbf{W}_1^T\mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x},j-1)+\mathbf{b}_{2L} & j=2L \end{cases} \tag{4.4}$$

$$\underset{\substack{\mathbf{W}_1,\ldots,\mathbf{W}_L,\\ \mathbf{b}_1,\ldots,\mathbf{b}_{2L}}}{\arg\min} \frac{1}{2m}\sum_{i=1}^{m}\left\{\left\|\mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x}_i,2L)-\mathbf{x}_i\right\|_2^2\right\} \tag{4.5}$$

4.3.3.5 Removing the decoding part

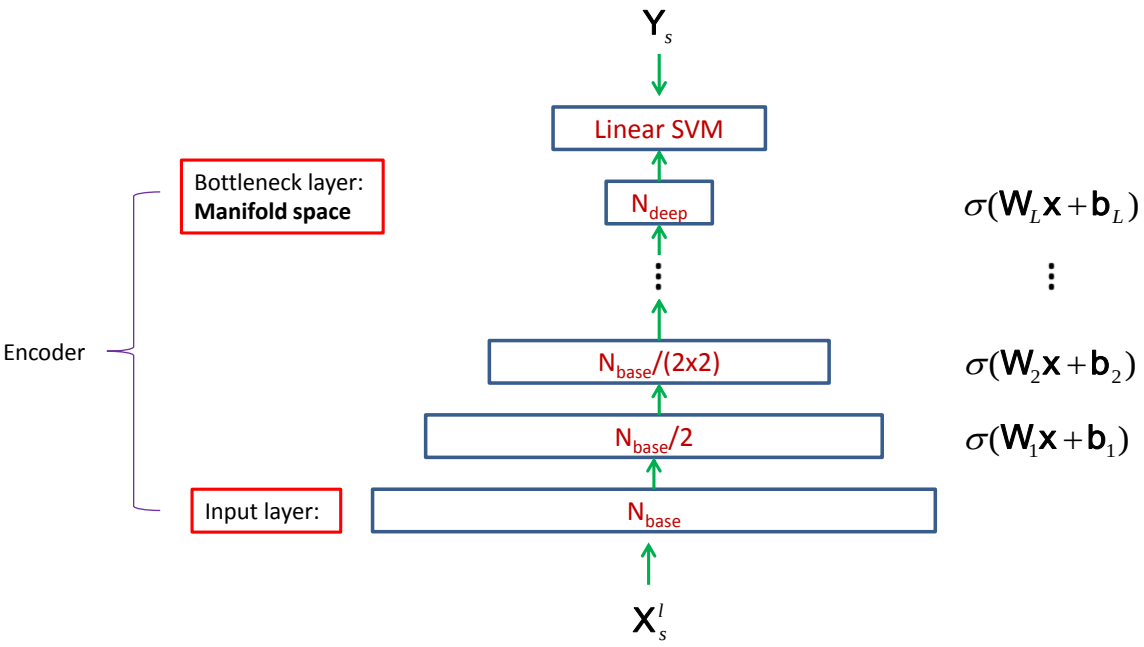


Figure 4.4: Training the scene-specific detector. The deep autoencoder has the decoder part removed. The source dataset $\mathbf{X}_s^l$ is projected to the manifold space and then a linear SVM is trained on the projected data and the corresponding class labels $\mathbf{Y}_s$.

After training the deep network, the decoder part is no longer needed and can thus be removed (*i.e.* only the first L network layers is retained). We now have a deep non-linear projection function $\mathcal{F}$ as given in Equation 4.6.

$$\mathcal{F}(\mathbf{x}) = \mathcal{A}(\mathbf{W}_1,\ldots,\mathbf{W}_L,\mathbf{b}_1,\ldots,\mathbf{b}_{2L},\mathbf{x},L) \tag{4.6}$$

where L is the number of projection layers (up to the bottleneck layer) and $\mathbf{x}$ is an input feature vector (*i.e.* base features) and $\mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, \mathbf{b}_{2L}$ are the learnt (*i.e.* optimized) network weights.

4.3.4 Training the Scene-specific Detector

We use the learnt encoder, $\mathcal{F}$, to project the generic dataset $\mathbf{X}_s^l$ and train a linear SVM[2] on these features and the corresponding source dataset labels $\mathbf{Y}_s$. This is illustrated in Figure 4.4.

4.4 Experimental Results

4.4.1 Datasets

INRIA pedestrian dataset [21] is used as the source dataset. We apply our algorithm and evaluate on two target video datasets: CUHK Square dataset [110] and MIT Traffic dataset [111]. Each of these videos were recorded by a static camera overlooking a far-field scene. Samples of frames from each video are shown in Figure 4.5 and details on the videos are given in Table 4.1. These datasets are challenging in that they vary significantly from the INRIA dataset in terms of resolution, camera angle, object poses and illumination conditions. Furthermore, they contain multiple categories of objects, and pedestrians in these datasets range from medium to low resolution.

Detail	CUHK Square	MIT Traffic
Frame size ($w \times h$)	720×576	720×480
Approximate video length (mins)	60	90
Frame rate (fps)	25	30
Number of frames	90,425	165,880

Table 4.1: Video dataset details

Each video dataset is divided into two halves:

- The 1st half is used for unsupervised detector adaptation. No manual annotations are used.

[2]... although any type of classifier can be trained. However, since the features are already highly non-linear, a further non-linear classifier is unlikely to be necessary and therefore a linear classifier is the best and the cheapest option.

Figure 4.5: Frame samples from the CUHK video (left) and from the MIT video (right). All frames are resized to the same aspect ratio for visualisation.

- The 2nd half for quantitative evaluation. For evaluation, the groundtruth that comes with the datasets [110, 111] are used. The groundtruth consists of manually annotated bounding boxes of pedestrians in the 100 uniformly sampled frames.

4.4.2 Base Features

For base feature extraction, all patches are resized to 64×128 (the same size as used in [21]). Each patch is partitioned to 8×8 cells, gradient histograms (9-orientations) are extracted in the cells, histograms are four-way normalised using the neighbouring cells (as in [21]) and all the histograms are concatenated to form a feature vector. This means that the (original) dimensionality of the dataset (*i.e.* base features) is:

$$N_{\text{base}} = 16 \times 8 \times 9 \times 4 = 4608$$

4.4.3 Unlabelled Patches Sampled for Deep Learning

We set the number of patches sampled (using Algorithm 4.2) to $10,000$ for all experiments. We do not sample any more than this due to memory and computational issues. We have tried fewer samples, but have found that our method is not really sensitive to this as shown in Figure 4.6.

Figure 4.7 gives a qualitative comparison between proposed biased sampling and random sampling; using the biased sampling greatly increases the chances of pedestrians being included in the samples for subsequent deep learning.

4.4.4 Deep Learning Parameters

The design parameters of the deep learning architecture that have been automatically computed for the CUHK Square and MIT Traffic datasets are summarised in Table 4.2. For training, mini-batch size is fixed at 1000 for both CUHK and MIT datasets.

The decrease in the optimization (loss/error) function value as the iterations increase is shown in Figure 4.8.

The entire adaptation system takes around 5 hours to run, with the majority of the time taken up by feature learning.

4.4.5 Implementation Details

The algorithms (including building the deep neural network architecture and training it) are mostly written and implemented using the MATLAB programming language, except

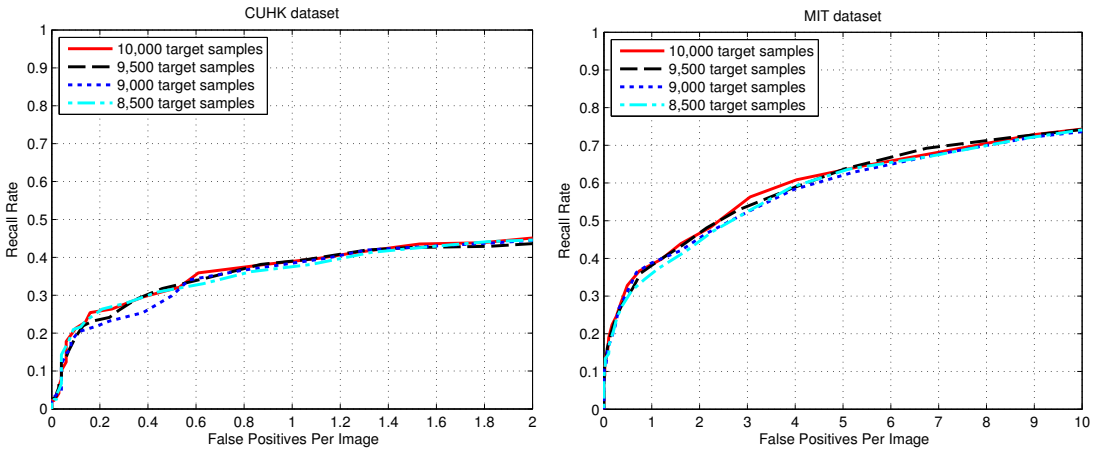


Figure 4.6: Effect of changing the number of samples from the target video on the (final) scene-specific detector performance. These graphs show that for both CUHK and MIT datasets, the domain adaptation algorithm is not sensitive to the number of target samples within the range tested (8,500-10,000). We have not tested fewer number of samples than 8,500 but it is to be expected that performance would start to decrease at some point (as the number of samples is reduced). Characterisation of this point is interesting in its own right and would require further study. Another observation that can be made from these graphs is that our algorithms is robust to the effect of different deep network initialisations.

	CUHK Square	MIT Traffic
Input dimensions (N_{base}). This is equal to the length of the input HOG feature vectors.	4608	4608
Bottleneck layer (*i.e.* manifold) dimensions (N_{deep}). This is automatically found using the method in [66].	35	23
Number of encoding hidden layers. Computed using the method proposed in Section 4.3.3.2.	5	6

Table 4.2: Deep learning architecture parameters

for some parts that are written in C/C++ and interfaced to MATLAB for speed. For training linear SVMs, LIBLINEAR [32] C++ library (with an interface to MATLAB) is used. For L-BFGS optimization, we use the *minFunc* Matlab function of Mark Schmidt [97]. The algorithms are implemented on a Intel Core i7 CPU (4 physical cores resulting in 8 logical cores) with 2.93 GHz processing power and 16 GB of RAM.

4.4.6 Evaluation & Discussion

We use recall-FPPI curves in order to compare the performance. Recall-FPPI curves have been commonly used and recommended in state-of-the-art literature to measure and

compare the performance of object detectors in images or videos [26, 110].

Detections (*i.e*. detected bounding boxes) are scored according to PASCAL 50% overlap criterion [31]. The overlap, α_o, is defined by:

$$\alpha_o = \frac{\text{area}(\mathbf{b}_g \cap \mathbf{b}_d)}{\text{area}(\mathbf{b}_g \cup \mathbf{b}_d)} \tag{4.7}$$

where $\mathbf{b}_g$ is the groundtruth bounding box and $\mathbf{b}_d$ is the detected bounding box. The overlap, α_o, gives the proportion of overlap between $\mathbf{b}_g$ and $\mathbf{b}_d$. A detection is deemed to be correct if $\alpha_o > 0.5$.

For each target dataset, we compare with three alternative methods as described below:

1. `Generic`: The detector (HOG+SVM) trained on the INRIA dataset. This is the baseline detector *without* any domain adaptation.

2. `PCA`: A simple detector adaptation method by applying Principal Component Analysis (PCA) on the combination of the source and target dataset. This experiment is done as a control to find out how well learning a simple linear subspace (using a well-known dimensional reduction technique) rather than a (deep) non-linear manifold (as proposed in this chapter) performs. The samples from the target video are obtained using our proposed biased sampling algorithm (*i.e*. Algorithm 4.2).

3. `Geodesic(CVPR12)`: The state-of-the-art approach proposed by Gong *et al*. [44]. However, since their method is only applicable for image classification, we extend the code made available by them for object detection in videos by using our proposed biased sampling algorithm.

4. `Proposed`: The detector obtained by our complete adaptation system.

We note that the goal of our proposed algorithm is *not* state-of-the-art (supervised) pedestrian detection. Instead, it is about state-of-the-art detector *adaptation*, *i.e*. how much the detector adaptation algorithm can improve over the baseline detector without any additional supervised knowledge from the target scene. The recall-FPPI curves are shown in Figure 4.9.

As can be seen, our algorithm (`Proposed`) significantly outperforms all three alternative methods, *i.e*. `Generic`, `PCA` and `Geodesic`, in both of the datasets.

For the CUHK square dataset, `Proposed` is much better than the baseline `Generic`, whereas `Geodesic` does not improve over the baseline. It turns out that `PCA` in fact performs worse than `Generic`.

For the MIT traffic dataset, both `Proposed` and `Geodesic` performs better than `Generic`, however `Proposed` has a significantly higher improvement than `Geodesic`. Here, `PCA` is also better than `Generic`, but only slightly.

From these experiments, the following observations can be made:

- For one of the datasets, the state-of-the-art `Geodesic` does not improve over the baseline whereas the proposed algorithm, due to unsupervised learning of deep non-linear features and the resulting implicit manifold regularization, achieves much better results in both of the datasets. This may also be the reason why `Proposed` outperforms `Generic` on both datasets.

- `PCA` performs either worse (for the CUHK dataset) or only slightly better (for the MIT dataset) than the generic detector, `Generic`. In contrast, `Proposed` significantly and clearly outperforms `Generic` in both of the datasets. This shows the advantages of learning a *nonlinear* manifold as proposed in this chapter.

- The state-of-the-art `Geodesic` is indeed better than `PCA` for both datasets. However, the difference between them is small. And it is interesting to note that both `Geodesic` and `PCA` learns linear subspaces and this may be part of the reason why their performances are quite close. It can also be observed that the state-of-the-art algorithm `Geodesic`, although optimizing a mathematical formulation of a common feature representation that is specially designed for domain adaptation, is not that much better than a simple and general `PCA`.

As mentioned in Section 4.3.4, due to the highly non-linear features (or manifold) learnt by the deep learning stage, it is sufficient to train a linear classifier in the manifold space. To verify that this is indeed the case, on the MIT Traffic dataset, we perform an additional experiment in which a Random Forest [12] (which is currently one of the most widely-used non-linear classifiers) is trained (instead of a linear SVM) in the manifold space. The result is illustrated in Figure 4.10. It can be seen that training a linear classifier is sufficient in the manifold space; in fact, training a Random Forest reduces the performance. This fits with the intuition that since the features projected onto the manifold space (which has been learnt in an unsupervised way) is already highly-nonlinear, attempting to make it even more non-linear with the limited amount of labelled data would most likely over-fit on the training data (*i.e.* the generic dataset), resulting in decreased performance (in the target dataset).

Figure 4.7: The effectiveness of the proposed biased sampling. The figure on the left shows a subset of patches sampled by the proposed biased sampling technique (Algorithm 4.2). On the right is a subset of samples obtained by (naive) random sampling.

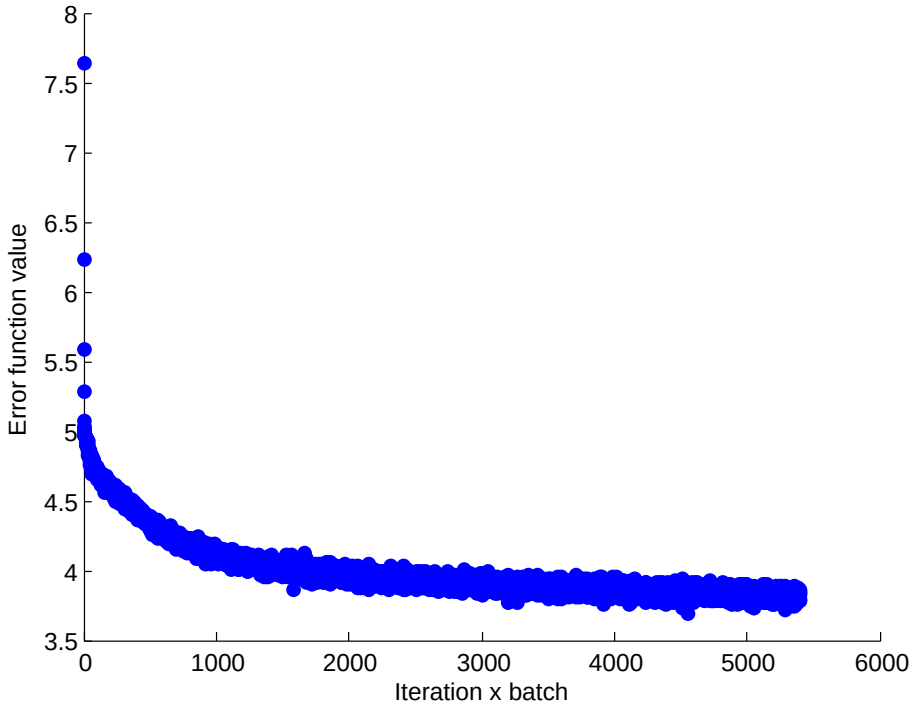
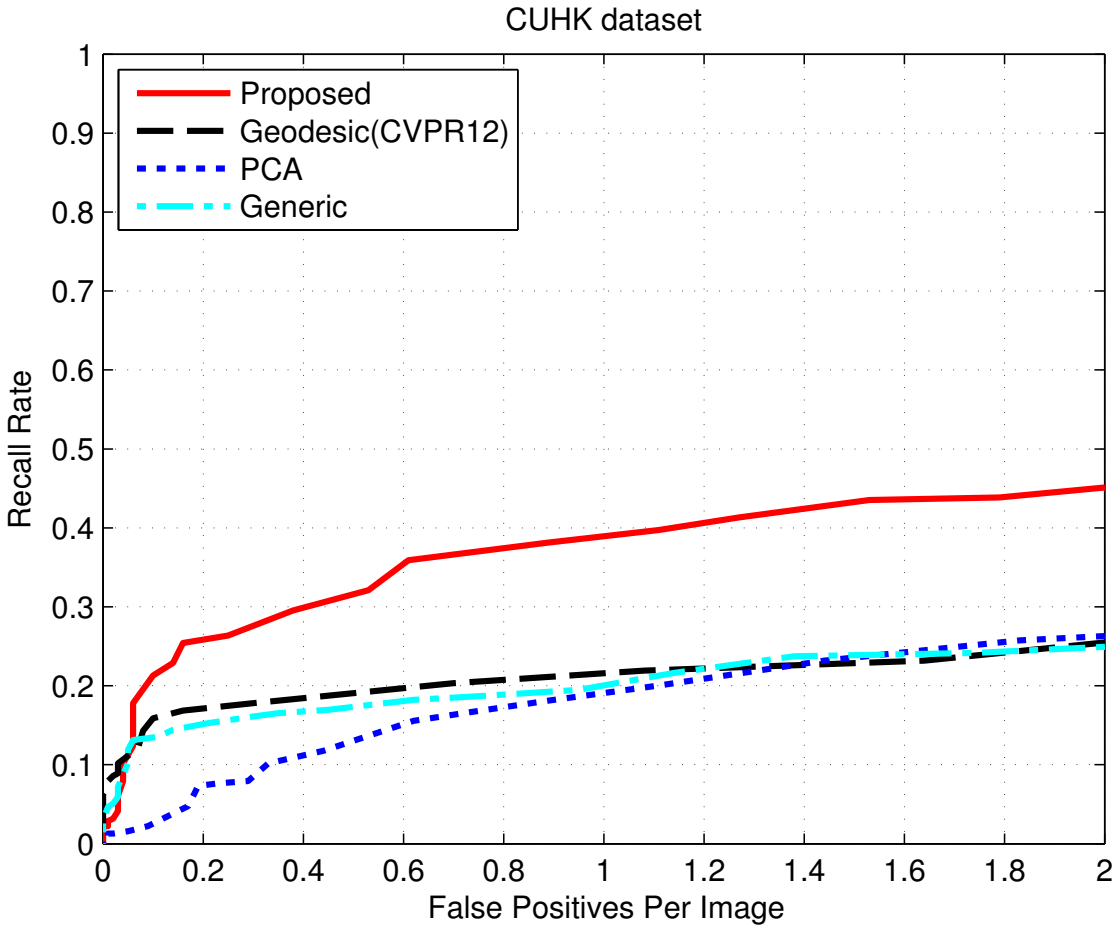


Figure 4.8: Objective error function value decreasing over iterations and converging. The noise is due to the stochastic nature of the mini-batch training. On the horizontal axis, "Iteration x batch" denotes "iteration multiplied batch" which refers the total number of mini-batch iterations. The number of iterations over the entire training dataset is given by the total number of mini-batch iterations $\times \frac{\text{Mini-batch size}}{\text{Data size}}$.



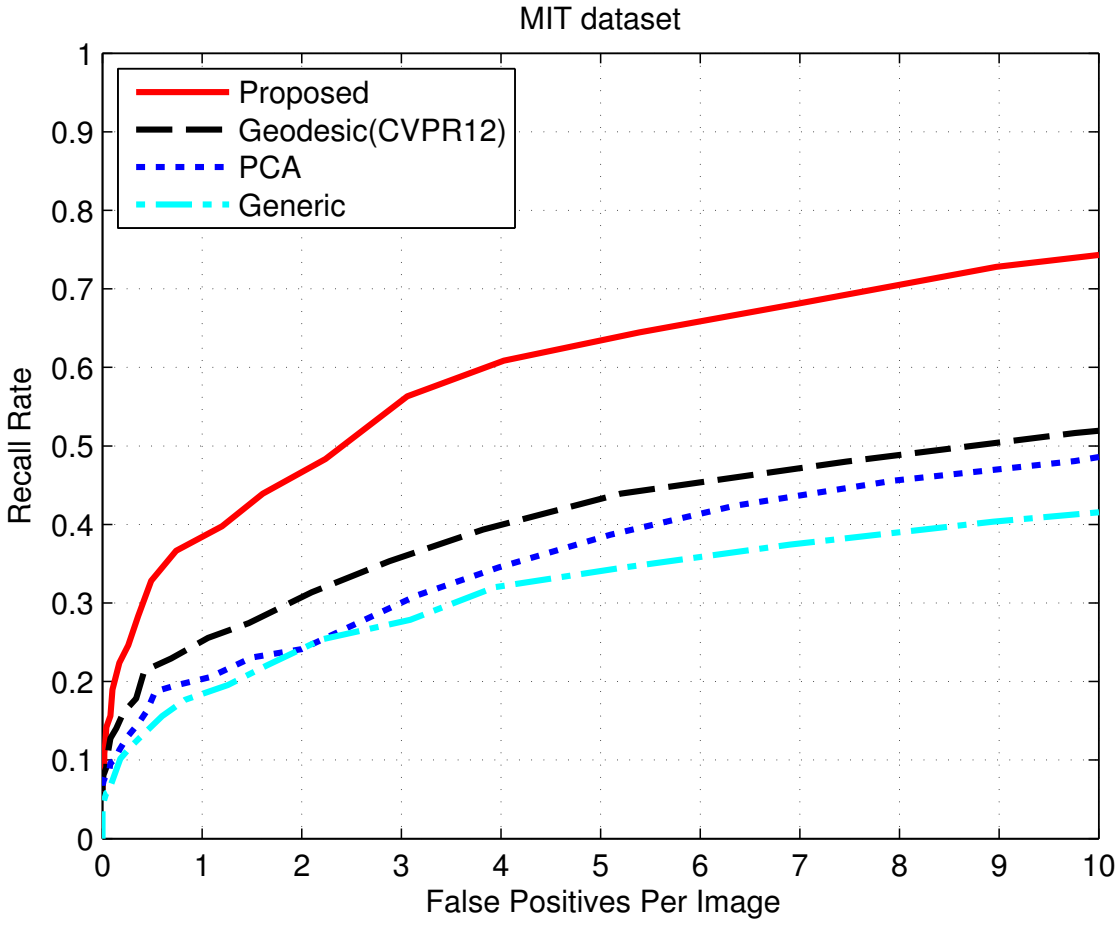


Figure 4.9: Detection recall-FPPI curves

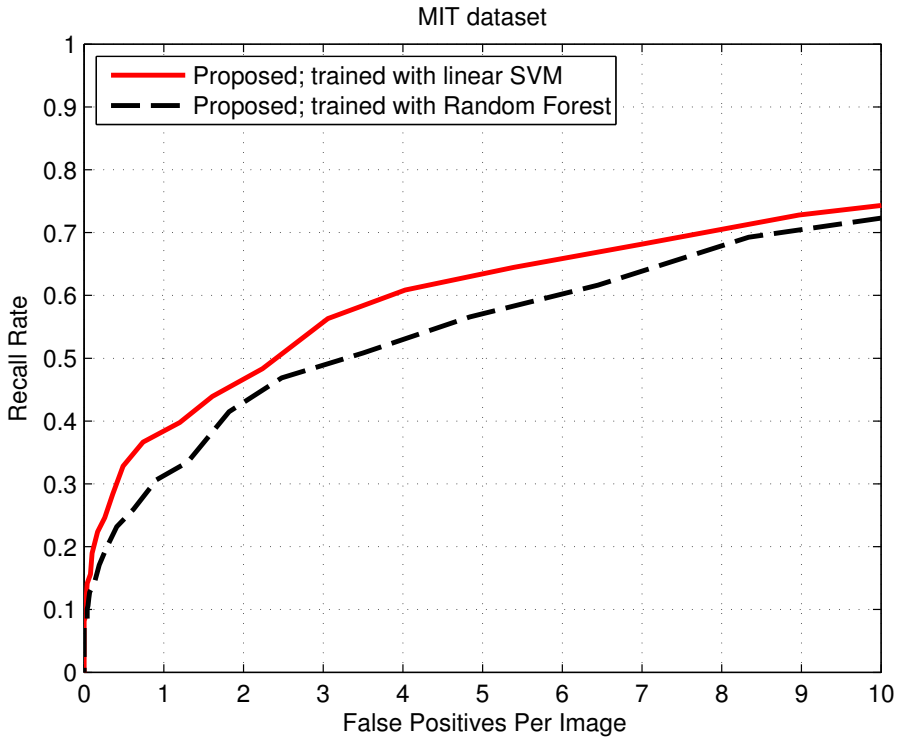


Figure 4.10: Effect of training with a Random Forest in the manifold space.

4.5 Conclusion

In this chapter, we propose an algorithm to automatically generate a scene-specific pedestrian detector that is tuned to a particular (video) scene by unsupervised domain adaptation of a generic dataset. Using state-of-the-art deep learning, 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. Quantitative evaluation on two publicly available video datasets show the effectiveness of our approach.

Although the algorithm proposed in this chapter significantly outperforms baselines and a state-of-the-art approach, there are further improvements that could be made:

1. A scene-specific detector is obtained by training in the manifold space that will perform well on both the source and the target dataset. However, this will not only expend its representational power in trying to do well on the target dataset, but also on the source dataset. It is reasonable to expect that a higher performance will result if the algorithm just focusses on doing well *only* on the target dataset without concerning about the source dataset. In Chapter 5, we propose an algorithm that generates a scene-specific detector that is tuned to perform well only on the target dataset.

2. At test time, there is a need to do feature projection before classification. This is time-consuming for object detection where hundreds of thousands of sliding windows need to be evaluated. The algorithm proposed in Chapter 5 overcomes this by not requiring any feature projection before classification.

3. The method in this chapter learns the manifold explicitly and therefore learns the similarity between data but it does not use cues and spatio-temporal information that are readily available in videos. These cues are effectively made use of by the algorithm proposed in Chapter 5. In fact, we will show that, by exploiting such powerful cues, we can forego the expensive manifold learning and feature projection steps and have a simple and efficient system that outperforms the method in this chapter.

Chapter 5

Efficient Non-iterative Domain Adaptation of Pedestrian Detectors

---

5.1 Overview

In this chapter, we present the second major contribution of the thesis; we propose a novel algorithm that makes effective and efficient use of cues available in video to automatically adapt and tune a generic pedestrian detector to a video that may possess different data distributions than the generic dataset from which the detector was trained. Most state-of-the-art approaches (in the related research area) can be inefficient, require manual setting of the number of iterations to converge and some form of human intervention. Our algorithm is a step towards overcoming these problems and although simple to implement, exceeds state-of-the-art performance.

5.2 Introduction

The high level view of this chapter is shown in Figure 5.1. We have a source dataset with supervision given in the form of pedestrian annotations and a target video dataset where supervision information is not available. The pedestrians in the target video have a different data distribution than the ones in the source dataset due to factors such as different poses, image resolution, camera angle and illumination conditions. The source dataset

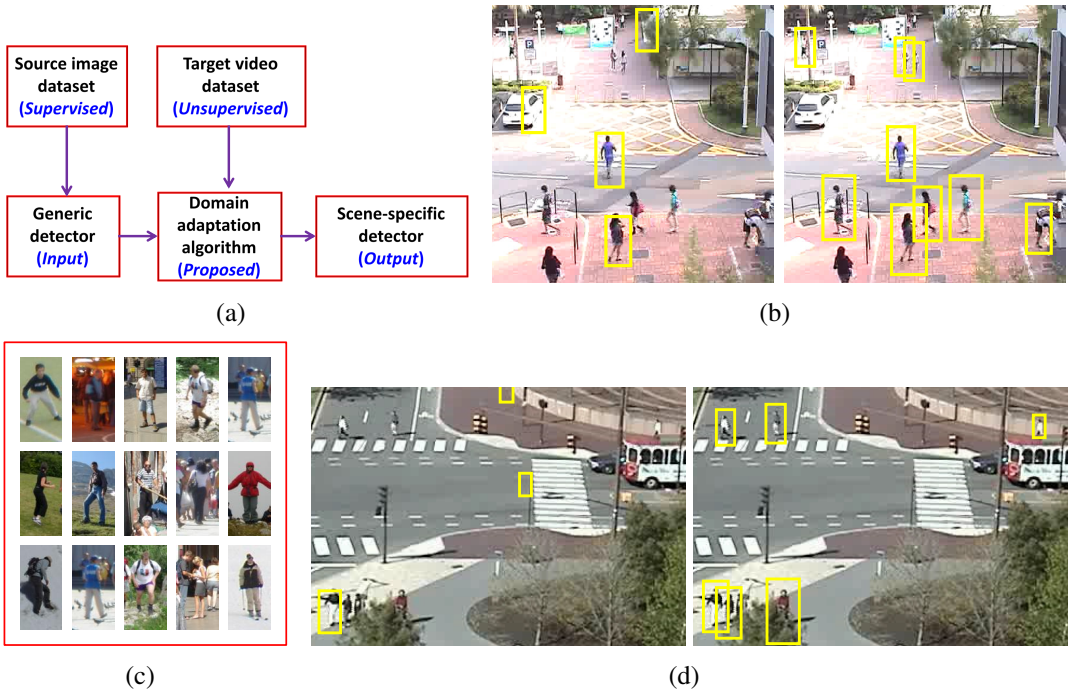


Figure 5.1: The high level view of this chapter. (a) shows the aim of this chapter. We have a generic detector trained on a generic dataset (c) available. The proposed algorithm automatically adapts it to different target scenes given only videos of those scenes without any label information attached with the videos. (b) & (d) show the improvement of the automatically adapted scene-specific detector over the generic detector for two target scenes. For (b) & (d), the left figure shows the detection results for the generic detector and the right for the adapted detector. For visualisation, all detection results shown are thresholded at around 1 False Positive Per Image (FPPI).

is a generic pedestrian dataset which is publicly available. The aim is to *automatically* generate a scene-specific detector which is tuned to the target video and would therefore perform better than the generic detector.

This domain adaptation problem may seem like an infeasible task since no supervision from the target video is available. However, this is not usually the case because there are certain assumptions we can make about the structure of the underlying distribution of the source data and target data [56]. In fact, in Chapter 4, we *explicitly* made use of the structure of the underlying distributions by learning the manifold jointly spanned by the source and target dataset. The algorithm proposed in that chapter could be applied to static images as well as videos. In this chapter, however, we develop an algorithm that is designed to work only with videos.

For videos, apart from the structural assumption about the data distributions, there is knowledge that can be exploited and is unique to videos such as the ability to model long-term scene background information from which to infer foreground objects and the

knowledge that objects move in a smooth and spatially and temporally coherent manner (which, for example, allows object tracking). In this chapter, instead of attempting to explicitly learn or model the manifold structures, our proposed algorithm makes effective use of this rich spatio-temporal "scene" knowledge and we show that it can render the task of detector adaptation for videos much simpler.

The rest of the chapter is organised as follows. In Section 5.3, we list the contributions of the chapter. In Section 5.4, we describe the overview and the details of the proposed algorithm. Section 5.5 discusses the experiments and the results. Finally, Section 5.6 gives the conclusion for the chapter.

5.3 Contributions

The contribution of this chapter is four-fold.

- Firstly, we introduce the idea of *bounding box proposals* and *initial verification* for efficient generation of a large number of scene-specific pedestrian data points with high probability of accuracy.
- Secondly, we use short-term tracking for spatio-temporal verification and *data expansion* (*i.e*. collection of hard pedestrian data).
- Thirdly, we show that this combination of bounding box proposal and initial verification, spatio-temporal verification and expansion does not need iterative self-training, effectively making it a *non-iterative* algorithm. Despite that, our algorithm can compete or outperform state-of-the-art iterative self-training algorithms.
- Fourthly, unlike most state-of-the-art algorithms (and the algorithm proposed in Chapter 4), the algorithm proposed in this chapter does not require the generic dataset for detector adaptation: just the generic detector alone is sufficient. This is useful in many situations. For example, the generic dataset may not be available for many reasons such as copyright issues. Or the generic dataset may be very large, making it difficult or costly to store and transmit for the purpose of domain adaptation, especially when there are many different target scenes for which detector adaptation needs to be performed.

5.4 Our Approach

5.4.1 Overview

The overview of our algorithm is illustrated in Figure 5.2. We describe the algorithm briefly below and the details of the algorithm are explained in the following sections.

The inputs to the algorithm are a generic (*i.e.* source) detector $\mathcal{C}_s$ and a target video $\mathbb{V} = [I_1, I_2, \ldots, I_N]$ of N frames to adapt $\mathcal{C}_s$ to. The desired output is a scene-specific detector $\mathcal{C}_t$ that is tuned to the target video.

The first step involves bottom-up generation of *bounding box proposals* of pedestrians for $\mathbb{V}$. Then these bounding box proposals are verified using C_s (*initial verification*). The result is a set of *verified proposals*. Each of these verified proposals is tracked for a short period (*e.g.* for 3 seconds). Then each track is verified using $\mathcal{C}_s$ and majority voting (*spatio-temporal verification*). For each verified track, the first data in the track and all the data that give negative labels (*i.e.* hard positives) are collected. The hard positives in a verified track can be considered as the *expansion* of the verified proposal that started the track.

All these *expanded* data are pooled across all the verified tracks to form the positive data for the scene-specific detector. Negative data for the scene-specific detector are sampled from the regions in $\mathbb{V}$ that $\mathcal{C}_s$ classifies as negative *and* that do not overlap with any areas of the bounding box proposals.

After the scene-specific positive and negative data have been obtained, the scene-specific detector $\mathcal{C}_t$ is trained on this data. The approach is formalised in Algorithm 5.1.

5.4.2 Generating Bounding Box Proposals

The first step of the algorithm is to propose bounding boxes of pedestrian candidates in a bottom up fashion. We do this by performing a *background subtraction* algorithm on $\mathbb{V}$ and *Connected Component Analysis* on the resulting foreground pixels and get tight bounding boxes around the connected components. These *bounding box proposals* across all the frames are *pooled* and stored in a set $\mathbb{B}$.

5.4.3 Initial Verification

We go through each bounding box proposal $\mathbf{b}_i \in \mathbb{B}$, get the image patch underlying $\mathbf{b}_i$, resize the patch and extract the features using the given feature extraction function, $\mathcal{H} : \mathbb{R}^{N_r \times N_c \times 3} \rightarrow \mathbb{R}^{N_{\text{base}}}$ where N_r and N_c are the number of rows and columns of the resized

Algorithm 5.1 Non-iterative detector adaptation using spatio-temporal cues

Input: $\{\mathcal{C}_s, \mathbb{V}\}$
Output: $\mathcal{C}_t$

% Generate bounding box proposals %
Bounding box proposals, $\mathbb{B} \leftarrow \varnothing$
Let Ω be an initial estimate of the scene background.
for $I_i \in \mathbb{V}$ **do**
  $\Omega \leftarrow \text{UpdateBGModel}(I_i, \Omega)$
  $I_{\text{fgmask}} \leftarrow$ Background subtraction using $\{I_i, \Omega\}$
  connected blobs $\leftarrow$ Connected Component Analysis on I_{fgmask}
  $\mathbb{B} \leftarrow \mathbb{B} \cup \{\text{bounding boxes of the connected blobs}\}$
end for
% Initial verification %
Verified proposals, $\mathbb{B}_v \leftarrow \varnothing$
Let $\mathcal{H}$ be the function for feature extraction
for $\mathbf{b}_i \in \mathbb{B}$ **do**
  if $C_s(\mathcal{H}(\mathbf{b}_i)) = 1$ **then**
    $\mathbb{B}_v \leftarrow \mathbb{B}_v \cup \{\mathbf{b}_i\}$
  end if
end for
% Spatio-temporal verification & expansion %
Scene-specific positive data, $\mathbf{X}_t^+ \leftarrow \varnothing$
for $\mathbf{v}_i \in \mathbb{B}_v$ **do**
  Set of tracked bounding boxes, $\mathbb{B}_e \leftarrow \text{ShortTermTrack}(\mathbf{v}_i)$
  Set of classification labels, $\mathbb{S} \leftarrow \varnothing$
  for $\mathbf{e}_j \in \mathbb{B}_e$ **do**
    $\mathbb{S} \leftarrow \mathbb{S} \cup \{\mathcal{C}_s(\mathcal{H}(\mathbf{e}_j))\}$
  end for
  if $\text{mode}(\mathbb{S}) = 1$ **then**
    for $\mathbf{e}_j \in \mathbb{B}_e$ **do**
      if $j = 1$ **or** $\mathcal{C}_s(\mathcal{H}(\mathbf{e}_j)) = 0$ **then**
        $\mathbf{X}_t^+ \leftarrow \mathbf{X}_t^+ \cup \{\mathcal{H}(\mathbf{e}_j)\}$
      end if
    end for
  end if
end for
% Collect scene-specific negative data %
$\mathbf{X}_t^- \leftarrow \varnothing$
for $I_i \in \mathbb{V}$ **do**
  Let $\mathbb{W}$ be the set of sliding windows on I_i
  $\mathbb{W} \leftarrow \{\mathbf{w} \in \mathbb{W} : (\mathcal{C}_s(\mathcal{H}(\mathbf{w})) = 0) \wedge (\mathbf{w} \cap \mathbb{B} = \varnothing)\}$
  $\mathbf{X}_t^- \leftarrow \mathbf{X}_t^- \cup \mathcal{H}(\mathbb{W})$
end for
% Train scene-specific detector %
$(\mathbf{X}_t^l, \mathbf{Y}_t) \leftarrow \{\mathbf{X}_t^+, \mathbf{X}_t^-\}$
$\mathcal{C}_t \leftarrow \text{LearnClassifier}(\mathbf{X}_t^l, \mathbf{Y}_t)$
return $\mathcal{C}_t$

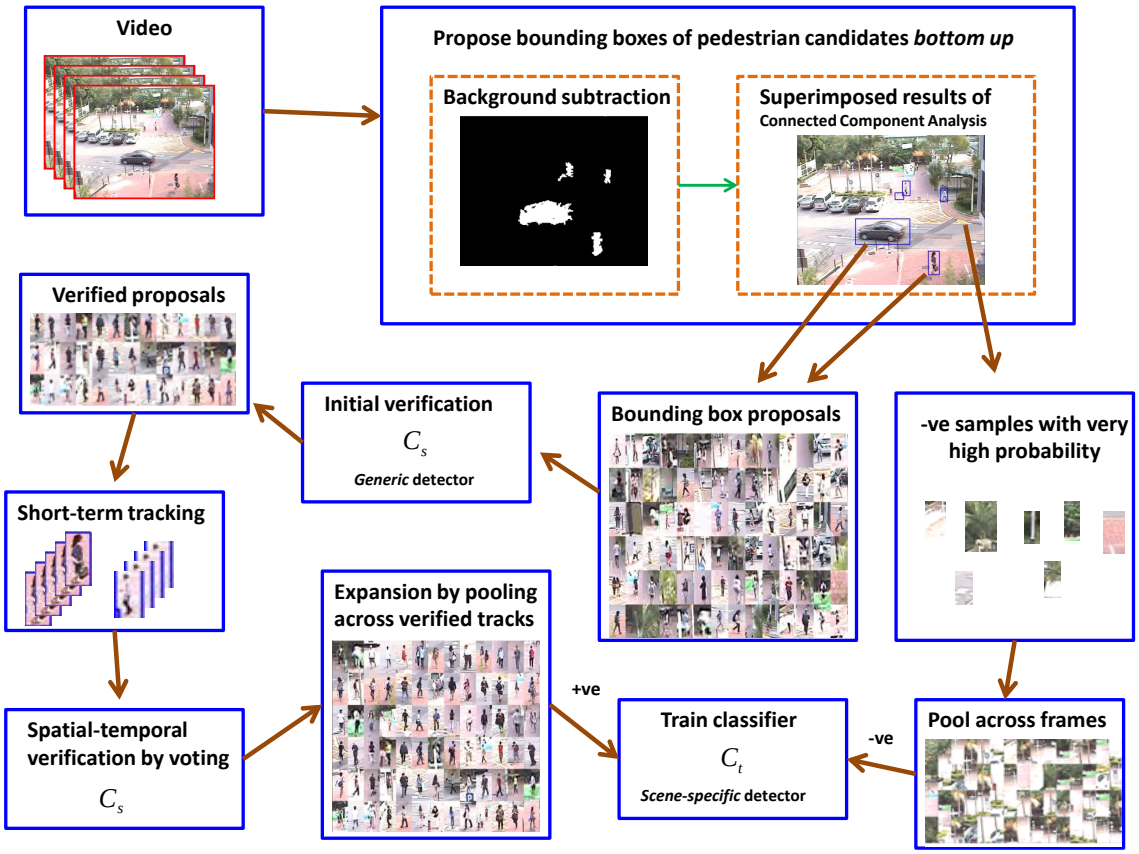


Figure 5.2: Overview of our proposed approach.

image patch and N_{base} is the length of the feature vector. These feature vectors are then passed to the generic detector, $C_s : \mathbb{R}^{N_{\text{base}}} \rightarrow \{1,0\}$, for labelling. If the label is 1 (*i.e.* the prediction is a pedestrian), $\mathbf{b}_i$ is considered a *verified proposal* and is stored in a set $\mathbb{B}_v$.

The combination of the background proposal and initial verification stages efficiently samples a high number of pedestrian patches with high probability. Random samples from sets $\mathbb{B}$ and $\mathbb{B}_v$ are shown in Figure 5.3. We can observe that $\mathbb{B}_v$ barely contains any mistakes. This is because the errors introduced by bounding box proposal and initial verification are mostly *uncorrelated*. In addition, $\mathbb{B}_v$ consists of very accurately localised pedestrians (i.e. very few patch alignment errors, etc.) which are suitable for training a detector.

5.4.4 Spatio-temporal Verification & Expansion

Although the set $\mathbb{B}_v$ is large and contains pedestrians with high probability, one might argue that $\mathbb{B}_v$ may be biased towards pedestrians that the generic detector is already "good" at and it might also contain some errors that the combination of the proposal generation

(a) (b)

(c) (d)

Figure 5.3: (a) & (b) correspond to the MIT Traffic dataset and (c) & (d) to the CUHK Square dataset. (a) & (c) show 200 random samples from set $\mathbb{B}$ and (b) & (d) show 200 random samples from set $\mathbb{B}_v$ (see Section 5.4 for notation).

and initial verification stages could not eliminate.

Therefore, we spatio-temporally verify and *expand* the set $\mathbb{B}_v$ by short-term tracking each verified proposal $\mathbf{v}_i \in \mathbb{B}_v$ producing a *tracklet* $\mathbb{B}_e$. The short-term tracking is done independently for each $\mathbf{v}_i$. We do not use $\mathcal{C}_s$ to help with the tracking; instead we use an online-learnt appearance model. Not using the $\mathcal{C}_s$ during tracking allows us to *decouple* the errors made by $\mathcal{C}_s$ and the tracker. Spatio-temporal verification is done by applying

$\mathcal{C}_s$ on each patch corresponding to a tracked bounding box $\mathbf{e}_j \in \mathbb{B}_e$ in the track and taking the majority vote of the classification labels $\mathbb{S}$.

If the majority vote is positive, then we consider the tracklet $\mathbb{B}_e$ as *verified* and add the data in the tracklet to the collection of scene-specific positive data $\mathbf{X}_t^+$. In order to avoid the number of data from getting too large, instead of adding every patch in a verified tracklet as new positive data, we add only the hard positives, *i.e*. patches in the track that have classification labels equal to 0. Figure 5.4 illustrates the idea.

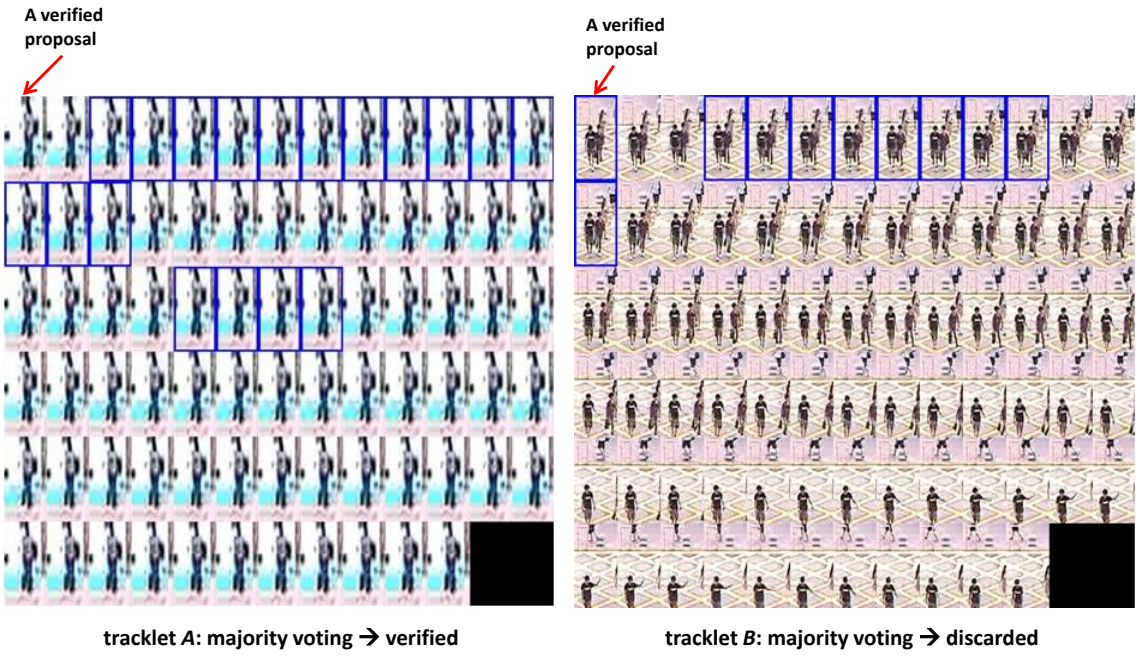


Figure 5.4: Visualization of two example tracklets obtained from tracking two verified proposals respectively. In each tracklet, we show the patches corresponding to the tracked bounding boxes. Since each tracklet is about 3-seconds long (76-frames), there are 76 patches in a tracklet (with the first patch being the verified proposal). For tracklet *A*, the blue rectangles indicate patches that the generic detector $\mathcal{C}_s$ classifies as non-pedestrians, *i.e*. they are false negatives. However, spatio-temporal verification (by majority voting in the track) successfully verifies the track as a pedestrian track. The patches with blue rectangles are therefore "hard" positives and are collected, along with the verified proposal, as scene-specific positive data $\mathbf{X}_t^+$. The patches with the blue rectangles can be termed as the *expansion* of the verified proposal. On the other hand, spatio-temporal verification on tracklet *B* correctly discards the track. Even though the first patch (the verified proposal) and other patches with the blue rectangles are erroneously classified by $\mathcal{C}_s$ as pedestrians (when in fact, each has 2 pedestrians in it), majority voting successfully discards the entire track along with the verified proposal.

5.4.4.1 Details on short-term tracking

The short-term tracking is performed by learning a discriminative model (*i.e.* classifier) of the appearance of the object being tracked against the surrounding patches (including the scene background and other objects). Different combinations of features and classifiers have been tried and the best one was found to be a combination of raw colour pixel values as features and Partial Least Squares (PLS) regression [40] as the classifier.

PLS regression combines the benefits of PCA and multiple regression. Given input and output matrices, **A** and **B** respectively, PLS simulteneously decomposes **A** and **B** into latent structures (to be more specific, linear subspaces) and these latent structures are jointly optimized such that the latent structure of **A** best explains the covariance between **A** and **B**.

In fact, by qualitative evaluation (*i.e.* visual inspection of tracking outputs), we found this combination (*i.e.* raw pixel values and PLS regression) to outperform several other trackers including:

- A tracker using online PCA on pixel values to learn and update an appearance model.
- Discriminative trackers using various types of features (such as HOGs and pixel intensity values) and SVM.
- Different types of classifiers for the discriminative tracker including online Perceptron and online logistic regression.

Since tracking is just a tool used as part of the domain adaptation algorithm and not in itself a novel contribution for this chapter, we do not present any further details on this.

5.4.5 Collecting Negative Examples

Scene-specific negative data, $\mathbf{X}_t^-$, are randomly sampled from all possible multi-scale sliding window bounding box regions that neither overlap with the bounding box proposals nor with pedestrian detections in the frames of the video. This way, negative data with high confidence can be obtained.

5.4.6 Training the Scene-specific Detector

Now that we have collected positive and negative scene-specific data ($\mathbf{X}_t^+$ and $\mathbf{X}_t^-$ respectively), we obtain a scene-specific detector $\mathcal{C}_t$ by training a classifier on the labelled target

dataset $(\mathbf{X}_t^l, \mathbf{Y}_t)$ which is the combination of $\mathbf{X}_t^+$ and $\mathbf{X}_t^-$. It should be noted that $\mathbf{X}_t^l$ is entirely made up of data from the target scene only and by doing this, we are effectively setting the weights for the source dataset to zero. We show the effectiveness of throwing away the source dataset in Section 5.5.

One of the strengths of our method is apparent here: we can use any classifier to train on $(\mathbf{X}_t^l, \mathbf{Y}_t)$. Thus, our approach is general and not limited to a particular type of classifier for the detector adaptation.

5.4.7 Analysis on Bounding Box Proposal & Verification

As discussed previously, the combination of bounding box proposal generation and initial verification are the first and second steps of our algorithm respectively. We now focus on this combination and compare it to state-of-the-art research [75, 110, 111] that uses background subtraction (as one of the steps) in the verification of detections of the generic detector $\mathcal{C}_s$. Their methods are different from our approach because we do *not* use background subtraction as a *verifier*; instead we use it the other way round: $\mathcal{C}_s$ is *the* verifier of background subtraction proposals.

Although this is a simple modification, it makes a significant difference in performance as experimentally shown in Section 5.5. There are several reasons on why our approach is preferable:

- We are not using $\mathcal{C}_s$ as a sliding window "detector", but we are making its task easier by using it as a classifier on the verified proposals. This minimises possible errors introduced by a sliding window detector (such as sliding window space discretization error and Non-maximum Suppression error).

- The intuition is to let the bottom-up process (*i.e.* bounding box proposal generation) "exert effort" and speak for itself as much as possible. Once it has fully "expressed" itself by going all the way to proposing bounding boxes, we then simply verify these proposals using $\mathcal{C}_s$. This eliminates requirement of any thresholds.

- Furthermore, if we use a real-time state-of-the-art background subtraction algorithm for bounding box proposal generation, we can very quickly and efficiently obtain a large number of verified proposals.

- Our approach is highly robust to errors in background subtraction and can work with videos and scenes where there are multiple categories of objects. This is because the *peculiarities* introduced by the combined process of background subtraction

(including shadow suppression) and Connected Component Analysis are mostly independent from the peculiarities of $\mathcal{C}_s$. Even assuming noisy background subtraction and Connected Component Analysis results, there will always be thousands of instances (out of hundreds of thousands of noisy instances) where the combined background subtraction and Connected Component Analysis will successfully yield correctly localised pedestrians. The generic detector $\mathcal{C}_s$ would correctly identify them with high probability and correctly reject others also with high probability.

It should be also noted that this is *not* the same as the approach taken in hierarchical segmentation-based selective search strategy to speed up object detection (*e.g.* [107]). The difference is that their goal is improving the detector per-se whereas our goal is domain adaptation and *not* to detect every single pedestrian (i.e. high recall with high precision) *during* the adaptation stage (instead, part of our goal during the adaptation stage is just to collect a reasonable amount of confident pedestrians). Furthermore, they are working on static images and hierarchical segmentation based on colour and texture cues whereas in our algorithm, we are directly using cues from the video without any hierarchical segmentation. Lastly, we do not apply any background subtraction or any other kinds of segmentation during test time after the detector adaptation.

5.4.8 Analysis on Spatio-temporal Verification & Expansion

There are methods in related work that use tracking-by-detection type of approach to improve detectors. This is typically done by applying the detector on every frame and then associating the detections into tracks. This can be considered as a process of temporal smoothing of detections to reduce noise. The problem, however, with this approach is that detection and tracking are dependent on each other and detection errors would be carried on to tracking and vice-versa. This can result in either drifting or the detector not improving. Therefore this type of tracking-by-detection is *not* a good oracle (*i.e.* verifier for self-training) because one of the requirements of the verification process is to be as independent from the detection as possible, while at the same time, offering useful (complementary) information.

Our method is different in that unlike (traditional) tracking-by-detection methods, our tracking is completely independent from detections of the generic detector since we do not associate frame-wise detections (from the generic detector) into tracks. Instead of relying on the detections (except for the initial detection that starts the track), our tracker makes use of the appearance (and shape) of the object being tracked to track the object. Only after completing the tracking, the detector is applied on instances (*i.e.* patches) in

the track and a majority vote is taken to verify the track. This allows the algorithm to decouple the errors of the detector and the tracker.

5.5 Experimental Results

5.5.1 Classifier & Features

For feature extraction and classification, we use Histogram of Oriented Gradients (HOGs) [21] and linear SVM respectively. This is done for simplicity and our algorithm can in principal be used with other feature extraction techniques and classifiers.

5.5.2 Datasets

As in Chapter 4, we use the INRIA Person dataset [21] as the source dataset, and MIT Traffic dataset [111] and CUHK Square dataset [110] as target datasets. Also similar to Chapter 4, we divide each video into two roughly equal parts:

1. 1^{st} half (*adaptation stage*): Used for (unsupervised) training. This is where all the detector adaptation takes place. No manual annotation is used for any detector adaptation.

2. 2^{nd} half (*testing stage*): After the detector adaptation is performed in the first half, the second half is used for testing. 100 frames are uniformly sampled and groundtruth is annotated for evaluation. It should be noted that in this stage, background subtraction, tracking or other cues are *not* used. Only pure (sliding window) detection performance is evaluated. The detector is applied independently on each frame being evaluated.

5.5.3 Descriptions of Experiments

Evaluation is performed in terms of recall-FPPI (False Positives Per Image) curves. The justification for using recall-FPPI curves for evaluation is that, in addition to the reasons given in Section 4.4.6 (of Chapter 4), recall-FPPI curves (especially recall at 1 FPPI) are used by the authors of the state-of-the-art algorithms that are being compared against our approach.

The PASCAL 50% overlap criterion [31] is used to score the detection bounding boxes. Six different types of experiments are performed:

1. `Proposed`: Our proposed algorithm (in this chapter).

2. `Baseline(Generic)`: The detector trained on the generic dataset. This is the baseline for our comparison.

3. `Nair (CVPR 04)`: This is an iterative self-training algorithm for detector adaptation using background subtraction. This can be considered as a variation of the approach of Nair and Clark [75].

4. `Wang (CVPR 11)`: A state-of-the-art detector adaptation algorithm that uses iterative self-training with multiple cues in the video [111].

5. `Wang (CVPR 12)`: Another state-of-the-art algorithm presented in [110] and is an extension of [111].

6. `Human supervision(`X`)`: Fully-supervised scene-specific detector obtained by manually annotation on X number of sampled frames in the 1^{st} half of video. Different values of X are used.

7. `Proposed + source dataset (INRIA)`: A modification of our proposed algorithm (`Proposed`). Instead of training the scene-specific detector only on the collected scene-specific data, we also include the source (*i.e.* generic) dataset for training.

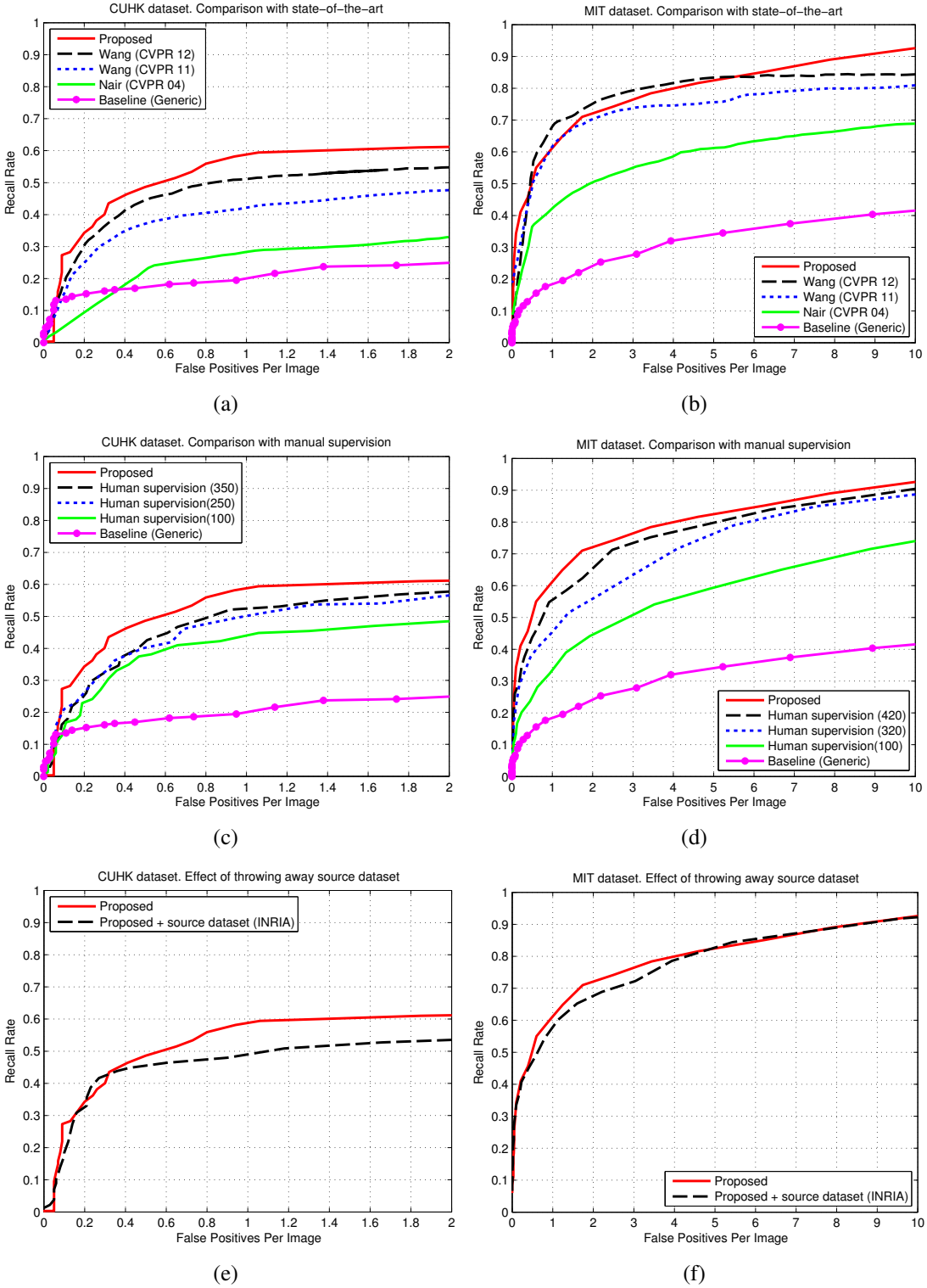


Figure 5.5: Detection performance curves (all on testing datasets). 1st column shows results for CUHK Square dataset. 2nd column shows results for MIT Traffic dataset. 1st row gives comparison of our proposed algorithm with state-of-the-art approaches. The 2nd row compares with manual annotation and the 3rd row shows the effect of throwing away the source dataset.

5.5.4 Evaluation

Performance curves are shown in Figure 5.5 with their plot legends referring to the types of experiments described previously. We discuss the plots below.

5.5.4.1 Comparison with generic detector

We see that our proposed method `Proposed` has a much higher performance than the generic detector `Baseline(Generic)` in all the experiments in both datasets. For CUHK Square dataset, at 1 FPPI, the recall of `Proposed` is about three times that of the generic detector, which is a significant improvement. For MIT traffic, the recall of `Proposed` is about 3.5 times higher than the recall of `Baseline(Generic)` at 1 FPPI. This shows that it is worthwhile to run the detector adaptation algorithm whenever we have a new scene and we want to automatically generate a much better detector than the generic detector.

5.5.4.2 Comparison with state-of-the-art

This is shown is Figure 5.5 (a) & (b) for CUHK and MIT datasets respectively. For CUHK dataset, our non-iterative algorithm `Proposed` clearly outperforms all the state-of-the-art iterative self-training approaches, `Wang(CVPR12)`, `Wang(CVPR11)` and `Nair(CVPR04)`, which require a manually set number of iterations to reach their peak performance given in the graphs.

For the MIT Traffic dataset, `Proposed` competes well with `Wang(CVPR12)` and is better than both `Wang(CVPR11)` and `Nair(CVPR04)`. Unlike in CUHK dataset, one of the reasons for `Proposed` not (clearly) outperforming `Wang(CVPR12)` may be that the MIT Traffic scene has more clearly defined and distinct paths for vehicles and pedestrians and since `Wang(CVPR12)` uses this information during the adaptation stage, their performance in this dataset is relatively better than their performance in the CUHK dataset. What is more interesting, however, is that even though our approach does *not* require and make use of this assumption (*i.e.* hand-labelling of pedestrian and vehicle paths), it can compete well with their approach which uses a lot of more cues and heuristics and has many more parameters to tune.

In both datasets, `Proposed` is significantly better than `Nair(CVPR04)` showing that our algorithm, despite using background subtraction in a major way, has a much higher performance due to it being a novel combination of bounding box proposal, initial verification, spatio-temporal verification and expansion by tracking.

5.5.4.3 Comparison with human supervision

The performance curves for detectors trained with varying amounts of human supervision is shown in Figure 5.5 (c) & (d). For CUHK, `Proposed` outperforms all the detectors trained with manual human supervision including the one that was trained with 350 frames worth of manual annotation. For MIT, similar observations can be made. However, as the number of frames that are manual annotated increases to a sufficient number, it is expected that `Human supervision(`X`)` may reach or exceed the performance of `Proposed`.

5.5.4.4 Effect of throwing away the source dataset

Our algorithm does not require the source dataset when training the scene-specific detector. The effect of including the source data is shown in Figure 5.5 (e) & (f). It can be observed that for both datasets, not incorporating the source dataset does *not* have any negative effect on the performance; in fact it even slightly improves the performance as compared to when incorporating the source dataset. This observation is consistent with the intuition that adding the source dataset is akin to trying to do well on *both* the generic dataset and target scene, with the net result that the scene-specific detector is less well tuned to the target scene.

5.5.4.5 Effect of training with another type of classifier

To show that our algorithm can naturally incorporate any type of classifier, we conduct an experiment in which the proposed domain adaptation algorithm is performed but this time, rather than training a linear SVM near the end of the domain adaptation, a Random Forest [12] is used. We demonstrate the result on the MIT Traffic dataset and is shown in Figure 5.6. It can be observed that the performance is worse when a Random Forest is used as the classifier. This may be because HOG features have a very high number of dimensions (and sparse) and Random Forests have been shown (*e.g.* in [98, 103]) to overfit for this type of data.

5.5.5 Analysis of Failure Modes

As discussed in Section 5.5.4.1, the scene-specific detector generated by our proposed domain adaptation algorithm significantly outperforms the generic detector and this shows the effectiveness of our proposed algorithm. Moreover, as discussed in Section 5.5.4.3, our domain adaptation algorithm even outperforms manual supervision with hundreds of

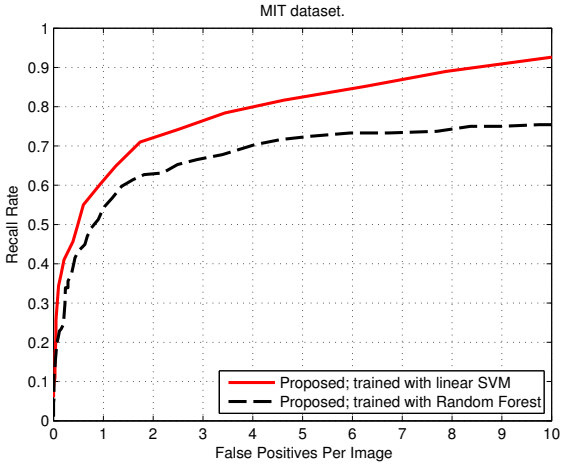


Figure 5.6: Effect of training with a Random Forest.

frames worth of manual supervision and Figures 5.5 (c) & (d) show that the performance of the manually supervised scene-specific detector as the number of manually supervised frames increases seems to be saturating.

These observations strongly suggest that with the proposed algorithm in this chapter, we have probably reached a point close to what is maximally achievable with the "power" of domain adaptation (*i.e.* the highest performance of what can be achieved with just domain adaptation given the same feature extraction and classifier type) because a scene-specific detector obtained by manual supervision gives the *upper bound* performance of the detector (whereas the generic detector gives the *lower bound* of what is achievable).

Furthermore, as has been noted a few times (including in Section 4.4.6), the scope of the thesis is *domain adaptation* (*i.e.* relative improvement of the adapted detector over the (original) generic detector given the *same* feature extraction mechanism and classifier type), *not* on absolute detection performance.

Therefore, it is then evident that any further shortcomings of the resulting scene-specific detectors (whether they are the outputs of our domain adaptation algorithms or they are trained with manual supervision in the target dataset) are mostly due to the inherent weaknesses associated with the combination of the feature extraction algorithm and the classifier type. And investigating this (*i.e.* pure pedestrian detection) is beyond the scope of this thesis.

However, to illustrate some representative detection failure cases (perhaps to serve as a motivation for researchers working in the field of pedestrian detection), we randomly sample some false positives and false negatives of the scene-specific detector and extract the patches corresponding to them and visualise them using *HOGgles* [109]. These are shown in Figures 5.7 and 5.8. As can be seen, most of the false positives actually do

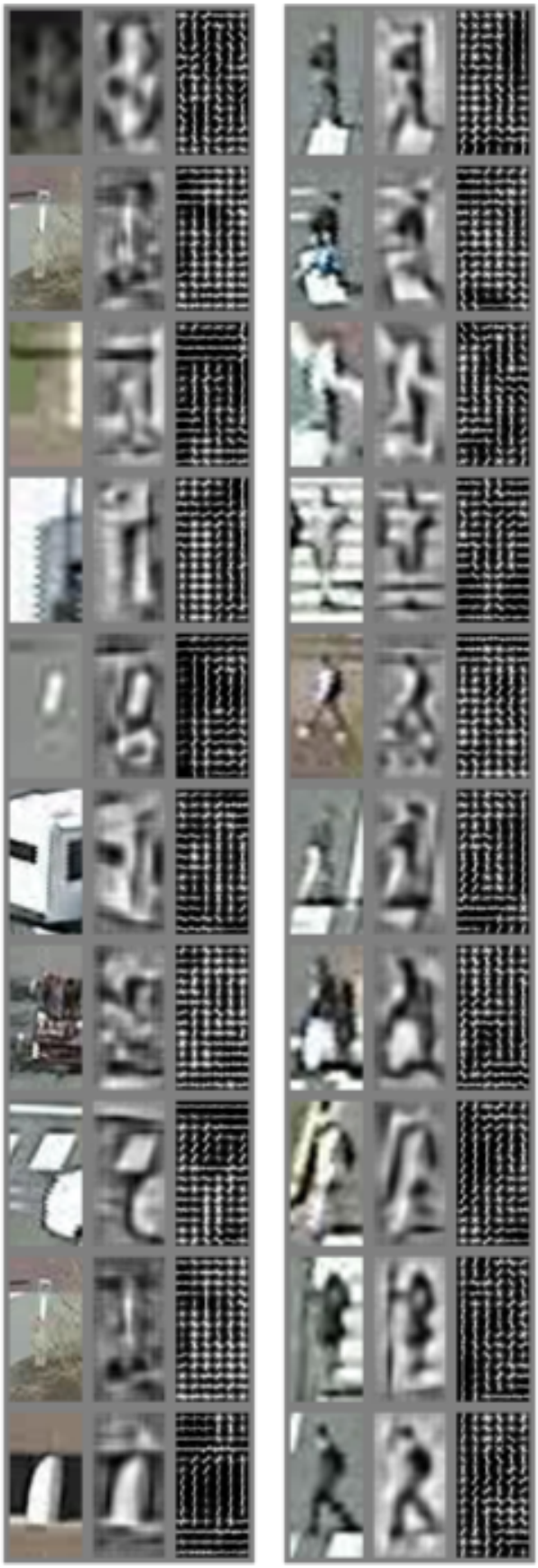
look like pedestrians in the HOG feature space and false negatives often do not resemble pedestrians in the HOG space.

The development of more powerful feature extraction algorithms and classifiers in the future will give rise to better pedestrian detectors; one of the advantages of our proposed domain adaptation algorithm is that it can naturally incorporate any combination of feature extraction and classifier type. And it can be expected that the proposed domain adaptation can then boost the performance of these more powerful generic pedestrian detectors to result in even better scene-specific pedestrian detectors. This is an open area of research.

Another open area of research is the domain adaptation of pedestrian detectors to scenes that are very crowded (such as shopping malls that are very busy). The domain adaptation algorithm presented in this chapter would not work well in this kind of scenario due to the overwhelming majority of the background proposals coming from wrongly connected foreground blobs and also because of short-term tracking errors in such highly crowded scenes. In addition, the proposed algorithm can also be sensitive to large camera movements.

(a) False positives (b) False negatives

Figure 5.7: Samples of false positives and false negatives in the CUHK Square dataset. Among the three columns, the first column represents the patch associated with the detection, the second column is the visualisation using *HOGgles* [109] and the third column shows the visualisation of HOGs.



(a) False positives (b) False negatives

Figure 5.8: Samples of false positives and false negatives in the MIT Traffic dataset. Among the three columns, the first column represents the patch associated with the detection, the second column is the visualisation using *HOGgles* [109] and the third column shows the visualisation of HOGs.

5.6 Conclusion

In this chapter, we propose an efficient and automatic non-iterative algorithm that adapts a generic detector to a specific scene given only the unlabelled video of the scene. The algorithm outputs a scene-specific detector that performs much better than the generic detector and performs as well as fully supervised detectors trained on hundreds of frames of manual annotations. Moreover, experimental results show that the algorithm outperforms state-of-the-art approaches on two challenging datasets. The scene-specific detector generated by our algorithm could be used as a building block for high level scene understanding or to improve tracking-by-detection applications.

Chapter 6

Weakly Supervised Detector Training by Unsupervised Prior Learning and Cue Fusion

6.1 Overview

The growth in the amount of collected video data in the past decade necessitates automated video analysis for which pedestrian detection plays a key role. Training pedestrian detectors using supervised machine learning techniques requires tedious manual annotation of pedestrians in the form of precise bounding boxes.

In the previous two chapters, we have investigated and proposed methods that adapt generic pedestrian detectors to specific videos. This has benefits in obtaining scene-specific pedestrian detectors with minimal human supervision by exploiting readily available generic datasets. In this chapter, we approach reducing human supervision from a different angle: we ask the question of reducing manual annotation effort when there are *no* such generic datasets to exploit.

We tackle this by proposing a novel weakly supervised algorithm to train pedestrian detectors for videos that requires annotations of only estimates of centres of pedestrian profiles (as shown in Figure 6.1) instead of the standard bounding box annotation used in most state-of-the-art research [5, 24, 25, 27, 29, 33, 42, 82]. This allows for an easier and

faster annotation compared to bounding box annotation.

Our algorithm makes use of a *pedestrian prior* learnt in an unsupervised way from the video and this prior is *fused* with the given weak supervision information in a principled manner.

We show on publicly available datasets that, despite the weak supervision, our algorithm performs comparably with bounding box supervision (termed in this chapter as *strong supervision*) despite having a much lower cost (measured in terms of the time it takes to complete the annotation). To be more precise, our weakly supervised algorithm reduces the cost of manual annotation by over four times while achieving similar performance as pedestrian detectors trained with bounding box annotations.

Figure 6.1: Strong versus weak annotation (best viewed in colour). On the left is the standard way of annotating pedestrians for training a pedestrian detector. On the right is the weak supervision (only approximate centres of pedestrians) required by our proposed algorithm. Note that pedestrians are of different sizes in the video due to projective distortion and hence our algorithm has to cope with *both* noisy locations and unknown scales. Weak supervision on the right is much faster and easier for a human annotator than the strong supervision on the left.

6.2 Contributions

We summarise the key contributions for this chapter as follows:

1. A weakly supervised training algorithm that makes use of approximate centre location annotation for training pedestrian detectors for videos.
2. Unsupervised learning of a pedestrian prior for a given video.
3. Combining cues from the unsupervised learnt prior and weak supervision in an optimization framework.
4. The algorithm works with low resolution videos that do not allow accurate part-based modelling and discovery, and that have multiple objects of varying sizes in each frame.
5. The algorithm is not sensitive to low-level segmentation unlike many state-of-the-art weak-supervision approaches using Multiple Instance Learning.
6. The approach is efficient since it does not require jointly solving all the weak supervisions.

6.3 Our Approach

An overview of the algorithm is illustrated in Figure 6.2. Let $\mathbb{V} = [I_1, I_2, \ldots, I_N]$ be a given video of N frames. Let $\mathbb{S}_{\text{weak}} = \{\mathbf{cc}_1, \mathbf{cc}_2, \ldots, \mathbf{cc}_M\}$ be a set of M given weak supervisions, where a weak supervision, $\mathbf{cc}_i = [cc_i^x, cc_i^y, cc_i^n]$, is a three element vector where cc_i^n gives the frame number in $\mathbb{V}$ associated with $\mathbf{cc}_i$, and cc_i^x and cc_i^y give the x-coordinate and y-coordinate (in image-plane) respectively corresponding to the approximate centre location of a pedestrian in frame cc_i^n of $\mathbb{V}$.

The goal is to obtain a pedestrian detector given only $\mathbb{S}_{\text{weak}}$ without being provided any bounding box annotations (which the traditional supervised training requires). Our algorithm is made up of three stages.

In the first stage, we learn a *pedestrian prior* in an unsupervised way using knowledge that can be automatically extracted from $\mathbb{V}$. In particular, for any video captured with a static uncalibrated camera, the dynamic background of the scene can be effectively modelled and foreground objects can be detected and extracted. Although these extracted foregrounds are usually very noisy, by letting the system observe a sufficiently long duration of $\mathbb{V}$, considering all the extracted foregrounds jointly and *generalising* over them

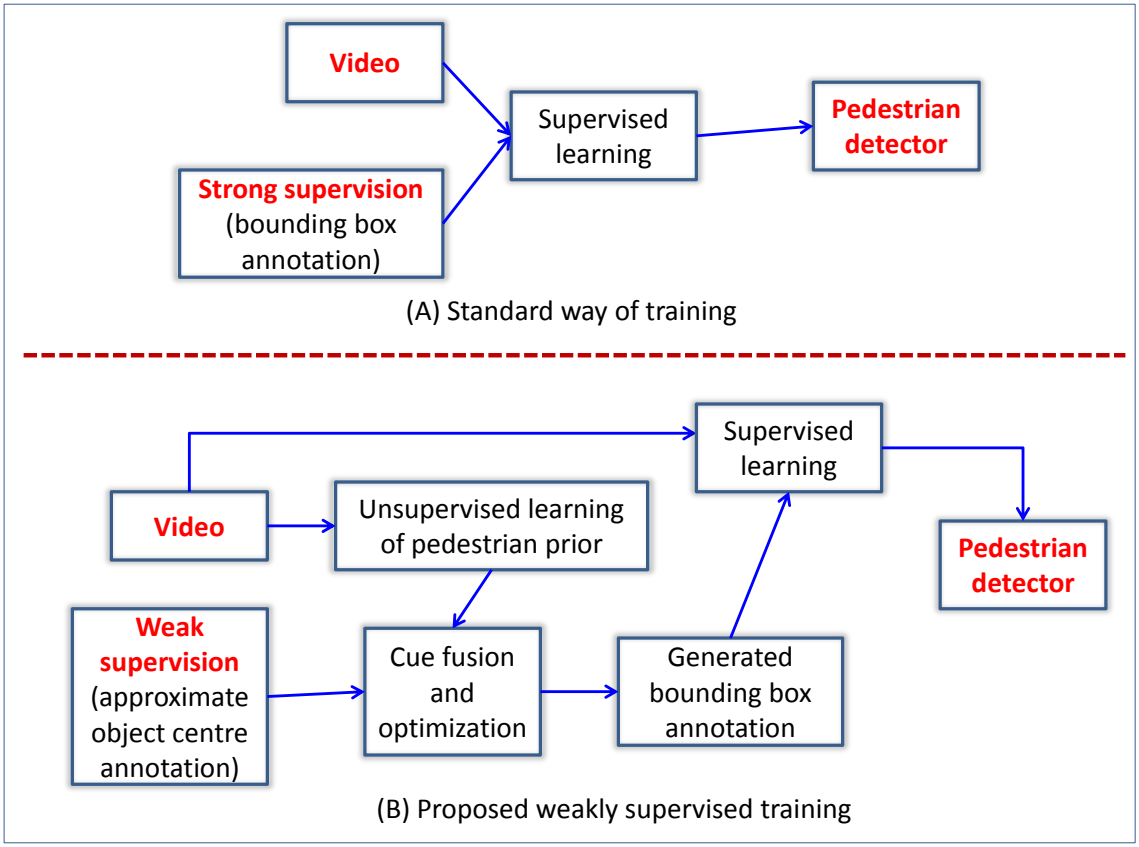


Figure 6.2: (A) shows the standard way of training pedestrian detectors. In comparison, (B) illustrates the overview of our proposed algorithm.

(in an offline process), the system can get a general idea and automatically learn about interesting objects in the scene associated with $\mathbb{V}$ without even attempting to explicitly distinguish between different object categories. This gives us a scene-specific *general object classifier* which could also be from another perspective be interpreted as a *pedestrian prior*.

More specifically, the pedestrian prior can be represented as $P(\text{pedestrian}|\text{patch})$, *i.e.* given any patch in $\mathbb{V}$, the pedestrian prior gives the *prior* probability[1] that the given patch depicts a pedestrian.

Due to noise and inaccuracies in the background modelling and subtraction process *and* due to the fact that this prior has been learnt using all classes of foreground objects in the scene, the pedestrian prior is error prone and would give high probabilities not only for pedestrians but also other object categories such as vehicles. However, we do not make any hard decisions at this stage and any errors and uncertainties in the pedestrian prior are resolved in the next stage using $\mathbb{S}_{\text{weak}}$.

[1] It is the probability or belief *before* seeing any (weak) supervision.

The second stage involves an optimization framework with an objective function that is a mixture of two terms:

1. The score of the pedestrian prior obtained in the first stage.

2. The agreement with the centres $\mathbb{S}_{\text{weak}}$.

We perform the optimization independently for each centre $\mathbf{cc}_i \in \mathbb{S}_{\text{weak}}$. Formulating in this way is very efficient compared to having to solve them jointly. After optimizing each weak supervision annotation (described in Section 6.3.2), we automatically obtain a bounding box annotation corresponding to the weak supervision.

Therefore, the second stage is in essence automatically converting the set of weak centre annotations $\mathbb{S}_{\text{weak}}$ to a set of bounding box annotations which are represented by $\mathbb{S}_{\text{bbox}} = \{\mathbf{bb}_1, \mathbf{bb}_2, \ldots, \mathbf{bb}_M\}$ where $\mathbf{bb}_i = [bb_i^{x_1}, bb_i^{y_1}, bb_i^{x_2}, bb_i^{y_2}, bb_i^{n}]$ is a 5-element vector with bb_i^n denoting the frame number in $\mathbb{V}$ associated with $\mathbf{bb}_i$, $bb_i^{x_1}$ and $bb_i^{y_1}$ giving the x and y coordinates (in image-plane) respectively of the upper left corner of the rectangle corresponding to the bounding box annotation and $bb_i^{x_2}$ and $bb_i^{y_2}$ representing the x and y coordinates of the bottom right corner of the bounding box.

After obtaining $\mathbb{S}_{\text{bbox}}$, we can now use any supervised learning algorithm to train a pedestrian detector. This is the third stage.

We formalise our approach in Algorithm 6.1 and explain it in detail in the coming sections.

6.3.1 Unsupervised Pedestrian Prior Learning

For a given video $\mathbb{V}$, a pedestrian prior, $\mathcal{C}_p : \mathbb{R}^{N_{\text{base}}} \rightarrow [0,1]$, is learnt in an unsupervised way, where N_{base} is the length of the feature vector input to the pedestrian prior and $\mathcal{C}_p$ outputs the (prior) probability that the given feature vector is a pedestrian. Let a feature extraction function be $\mathcal{H} : \mathbb{R}^{N_r \times N_c \times 3} \rightarrow \mathbb{R}^{N_{\text{base}}}$ where N_r and N_c are the number of rows and columns of an image patch and N_{base} is the length of the resulting feature vector.

The method for learning $\mathcal{C}_p$ is outlined in Algorithm 6.2 and we describe the steps of the algorithm below.

Firstly, background subtraction is performed for each frame $I_i \in \mathbb{V}$, followed by Connected Component Analysis (CCA). CCA gives a set of the bounding boxes corresponding to the connected components which are stored in a set $\mathbb{B}$.

Although any background subtraction technique could be used, since the unsupervised prior learning stage is offline and does not need real-time processing, a highly accurate and robust yet reasonably fast background subtraction algorithm (such as [116]) is practical.

Algorithm 6.1 Overview of the weakly supervised training

Input: Video $\mathbb{V}$ and weak supervision $\mathbb{S}_{\text{weak}}$
Output: Scene-specific pedestrian detector, $\mathcal{C}_t$

$\mathcal{C}_p \leftarrow \text{LearnPrior}(\mathbb{V})$, where LearnPrior is the function to learn unsupervised pedestrian prior. Described in Algorithm 6.2.

$\mathbb{S}_{\text{bbox}} \leftarrow \text{FuseOptimize}(\mathbb{V}, \mathbb{S}_{\text{weak}}, \mathcal{C}_p)$, where FuseOptimize is the function to convert weak centre supervision $\mathbb{S}_{\text{weak}}$ to bounding box annotations $\mathbb{S}_{\text{bbox}}$. Detailed in Algorithm 6.3.

$\mathbf{X}_t^+ \leftarrow \mathcal{H}(\text{patches corresponding to } \mathbb{S}_{\text{bbox}})$

$\mathbf{X}_t^- \leftarrow \varnothing$
for $I_i \in \mathbb{V}$ **do**
  Let $\mathbb{W}$ be the set of sliding windows on I_i
  $\mathbf{X}_t^- \leftarrow \mathbf{X}_t^- \cup \mathcal{H}(\{\mathbf{w} \in \mathbb{W} : \mathbf{w} \cap \mathbb{S}_{\text{bbox}} = \varnothing\})$
end for

$(\mathbf{X}_t^l, \mathbf{Y}_t) \leftarrow \{\mathbf{X}_t^+, \mathbf{X}_t^-\}$

$\mathcal{C}_t \leftarrow \text{LearnClassifier}(\mathbf{X}_t^l, \mathbf{Y}_t)$

return $\mathcal{C}_t$

Algorithm 6.2 Unsupervised pedestrian prior learning

Input: Video $\mathbb{V}$
Output: Unsupervised pedestrian prior $\mathcal{C}_p$

$\mathbf{X}_t^{\text{fg}} \leftarrow \varnothing$
$\mathbf{X}_t^{\text{bg}} \leftarrow \varnothing$

Let Ω be an initial estimate of the scene background.

for $I_i \in \mathbb{V}$ **do**

$\Omega \leftarrow \text{UpdateBGModel}(I_i, \Omega)$
$I_{\text{fgmask}} \leftarrow$ Background subtraction using $\{I_i, \Omega\}$
Connected Component Analysis on I_{fgmask}
$\mathbb{B} \leftarrow \{\text{bounding boxes of the connected blobs}\}$

for $\mathbf{b}_i \in \mathbb{B}$ **do**
$\mathbf{X}_t^{\text{fg}} \leftarrow \mathbf{X}_t^{\text{fg}} \cup \{\mathcal{H}(\mathbf{b}_i)\}$
end for

Let $\mathbb{W}$ be the set of sliding windows on I_i
$\mathbb{W} \leftarrow \{\mathbf{w} \in \mathbb{W} : \mathbf{w} \cap \mathbb{B} = \varnothing\}$
$\mathbf{X}_t^{\text{bg}} \leftarrow \mathbf{X}_t^{\text{bg}} \cup \mathcal{H}(\mathbb{W})$

end for

$(\hat{\mathbf{X}}_t^l, \hat{\mathbf{Y}}_t^l) \leftarrow \{\mathbf{X}_t^{\text{fg}}, \mathbf{X}_t^{\text{bg}}\}$
$\mathcal{C}_p \leftarrow \text{LearnClassifier}(\hat{\mathbf{X}}_t^l, \hat{\mathbf{Y}}_t^l)$
$\mathcal{C}_p \leftarrow$ Calibrate $\mathcal{C}_p$ to produce valid probabilities.

return $\mathcal{C}_p$

For each image patch corresponding to a bounding box $\mathbf{b}_i \in \mathbb{B}$, we compute features by applying the function $\mathcal{H}$ (after appropriate resizing of the patch). These feature vectors form a set $\mathbf{X}_t^{\text{fg}}$. The feature extraction function is general and any suitable method can be used. We use Histograms of Oriented Gradients (HOGs) features [21].

Random samples of patches from which $\mathbf{X}_t^{\text{fg}}$ is obtained are shown in Figures 6.3 and 6.4 for the CUHK Square and MIT Traffic datasets respectively.

Now, we take all the multi-scale sliding windows in each frame $I_i \in \mathbb{V}$ that do *not* overlap with any $\mathbf{b}_i \in \mathbb{B}$, resize and extract features using $\mathcal{H}$, giving the set of feature vectors $\mathbf{X}_t^{\text{bg}}$.

Finally, since we have collected both $\mathbf{X}_t^{\text{fg}}$ and $\mathbf{X}_t^{\text{bg}}$, we train a binary classifier $\mathcal{C}_p$ using $\mathbf{X}_t^{\text{fg}}$ as the positive class and $\mathbf{X}_t^{\text{bg}}$ as the negative class. This classifier (*i.e.* function) $\mathcal{C}_p$ is the pedestrian prior. The aim of $\mathcal{C}_p$ is simply to capture some information about the pedestrian class. $\mathcal{C}_p$ *implicitly* captures the multi-modal distribution about objects in the scene and obtaining $\mathcal{C}_p$ therefore does not require clustering and explictly discovering foreground object categories. $\mathcal{C}_p$ could be any type of classifier and in our case, a linear SVM is used.

If $\mathcal{C}_p$ does not output valid probabilities (such as when using an SVM), $\mathcal{C}_p$ should be calibrated to produce (proper) probabilities. One such way to calibrate it by using Platt scaling. It is essentially fitting a logistic regression on the classifier scores:

$$\mathcal{C}_p(\mathbf{x}) \leftarrow \frac{1}{1+\mathrm{e}^{-(\beta_0+\beta_1\mathcal{C}_p(\mathbf{x}))}} \tag{6.1}$$

where β_0 and β_1 are scalar parameters associated with the logistic regression and can be learnt from data and $\mathbf{x}$ is the feature vector of a data point that is input to the $\mathcal{C}_p$.

6.3.2 Cue Fusion and Optimization

The goal here is to fuse the unsupervised pedestrian prior $\mathcal{C}_p$ obtained (as described in the previous section) with the provided weak supervisions $\mathbb{S}_{\text{weak}}$ and optimize the combination of these two sources of information to generate bounding box annotations $\mathbb{S}_{\text{bbox}}$. The technique is formally given in Algorithm 6.3 and explained below.

The set of generated bounding box annotations $\mathbb{S}_{\text{bbox}}$ is initialised to an empty set $\varnothing$. There are M weak supervisions (*i.e.* $|\mathbb{S}_{\text{weak}}| = M$) and each weak supervision $\mathbf{cc}_i \in \mathbb{S}_{\text{weak}}$ is optimized independently. Therefore, below, we detail the process of converting a single weak supervision $\mathbf{cc}_i \in \mathbb{S}_{\text{weak}}$ to a single bounding box annotation $\mathbf{bb}_i \in \mathbb{S}_{\text{bbox}}$.

Firstly, we compute a large rectangular region Φ surrounding and centred at the weak supervision $[cc_i^x, cc_i^y]$. This can be computed by setting for the whole video, *estimates* of the widths and heights of the smallest and largest possible pedestrian in the scene. These do not need to be accurate, can be easily determined by a human and need to be set only once at the beginning of the algorithm.

Then, a set of K multi-scale sliding windows $\mathbb{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ are generated within Φ (and only windows whose widths and heights are larger than the smallest width and height set previously are retained). Each $\mathbf{w}_j \in \mathbb{W}$ is a bounding box and is defined by a vector $\mathbf{w}_j = [w_j^{x_1}, w_j^{y_1}, w_j^{x_2}, w_j^{y_2}]$.

After generating $\mathbb{W}$, we are now ready to fuse the unsupervised pedestrian prior $\mathcal{C}_p$ with the weak supervision to obtain a bounding box annotation. This is done in an opti-

Algorithm 6.3 Cue Fusion and Optimization

Input: Video $\mathbb{V}$, weak supervision $\mathbb{S}_{\text{weak}}$ and unsupervised pedestrian prior $\mathcal{C}_p$
Output: Bounding box annotations $\mathbb{S}_{\text{bbox}}$

$\mathbb{S}_{\text{bbox}} \leftarrow \varnothing$

Let $\{w_{\min}, w_{\max}, h_{\min}, h_{\max}\}$ be estimates of minimum and maximum possible widths and heights of pedestrians in $\mathbb{V}$.

for $i = 1$ **to** M **do**

% Get info from the current weak supervision $\mathbf{cc}_i \in \mathbb{S}_{\text{weak}}$ %
$\mathbf{cc}_i = [cc_i^x, cc_i^y, cc_i^n]$

% Get a large enough rectangular region surrounding current weak supervision %
$\Phi \leftarrow [cc_i^x - w_{\max}/2, cc_i^y - h_{\max}/2, cc_i^x + w_{\max}/2, cc_i^y + h_{\max}/2]$

$\mathbb{W} \leftarrow$ get multiscale sliding windows, larger than $w_{\min}$ and $h_{\min}$, in the region Φ. $\mathbb{W}$ is a set of K bounding boxes specified by $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ where $\mathbf{w}_j = [w_j^{x_1}, w_j^{y_1}, w_j^{x_2}, w_j^{y_2}]$ is a vector denoting the x and y coordinates of the top-left ($w_j^{x_1}$ and $w_j^{y_1}$) and the bottom-right ($w_j^{x_2}$ and $w_j^{y_2}$) corners of the bounding box $\mathbf{w}_j$.

Let $\mathcal{N}(\mathbf{w})$ be a function, $\mathcal{N} : \mathbb{R}^4 \rightarrow \mathbb{R}$, given by: $\mathcal{N}(\mathbf{w}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathcal{G}(\mathbf{w}) - \mu)^T \Sigma^{-1} (\mathcal{G}(\mathbf{w}) - \mu)\right)$ where $\mu = [cc_i^x, cc_i^y]$, $\Sigma = \left[\begin{smallmatrix} 3 & 0 \\ 0 & 3 \end{smallmatrix}\right]$ and $\mathcal{G}(\mathbf{w}) = \left[\frac{w^{x_1} + w^{x_2}}{2}, \frac{w^{y_1} + w^{y_2}}{2}\right]$.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w} \in \mathbb{W}} \frac{\mathcal{C}_p(\mathcal{H}(\mathbf{w}))}{\sum_{\mathbf{w} \in \mathbb{W}} \mathcal{C}_p(\mathcal{H}(\mathbf{w}))} + \frac{\mathcal{N}(\mathbf{w})}{\sum_{\mathbf{w} \in \mathbb{W}} \mathcal{N}(\mathbf{w})}$$

$\mathbf{bb}_i \leftarrow [\hat{w}^{x_1}, \hat{w}^{y_1}, \hat{w}^{x_2}, \hat{w}^{y_2}, cc_i^n]$ where $\mathbf{bb}_i \in \mathbb{S}_{\text{bbox}}$

end for

return $\mathbb{S}_{\text{bbox}}$

mization framework for which the objective function is essentially a combination of two terms:

1. The appearance term provided by the pedestrian prior $\mathcal{C}_p$: for a candidate window $\mathbf{w} \in \mathbb{W}$, the appearance term determines how likely the image patch corresponding to $\mathbf{w}$ is a pedestrian.

2. The spatial-scoring term provided by the weak supervision: for a candidate window $\mathbf{w} \in \mathbb{W}$, the spatial term gives the closeness of the centre of $\mathbf{w}$ to the weak supervision centre $[cc_i^x, cc_i^y]$.

We seek the best window $\hat{\mathbf{w}} \in \mathbb{W}$ such that $\hat{\mathbf{w}}$ is scored highest by the combination of these terms in the objective function as given below:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w} \in \mathbb{W}} \frac{\mathcal{C}_p(\mathcal{H}(\mathbf{w}))}{\sum_{\mathbf{w} \in \mathbb{W}} \mathcal{C}_p(\mathcal{H}(\mathbf{w}))} + \frac{\mathcal{N}(\mathbf{w})}{\sum_{\mathbf{w} \in \mathbb{W}} \mathcal{N}(\mathbf{w})} \tag{6.2}$$

The function $\mathcal{N} : \mathbb{R}^4 \rightarrow \mathbb{R}$, gives the likelihood of a window $\mathbf{w}$ to be consistent with the weak supervision and is defined as:

$$\mathcal{N}(\mathbf{w}) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\mathcal{G}(\mathbf{w}) - \mu)^T \Sigma^{-1} (\mathcal{G}(\mathbf{w}) - \mu) \right) \tag{6.3}$$

where $\mu = [cc_i^x, cc_i^y]$, $\Sigma = \left[\begin{smallmatrix} 3 & 0 \\ 0 & 3 \end{smallmatrix}\right]$ and $\mathcal{G}(\mathbf{w}) = \left[\frac{w^{x1} + w^{x2}}{2}, \frac{w^{y1} + w^{y2}}{2} \right]$. $\mathcal{G}$ is a function, $\mathcal{G} : \mathbb{R}^4 \rightarrow \mathbb{R}^2$, to get the centre coordinates of a bounding box $\mathbf{w}$.

The terms in the denominator of the optimization function are normalisation terms to make sure the relative weighing of the two terms in the objective function are equal.

Informally, the optimization objective prefers $\mathbf{w} \in \mathbb{W}$ that is scored highly by $\mathcal{C}_p$ but is penalised the further the $\mathbf{w}$ is from the weak supervision centre $[cc_i^x, cc_i^x]$.

The function $\mathcal{N}$ is actually a bivariate Gaussian distribution with the mean μ at the weak supervision centre and the covariance Σ is fixed a priori. The function allows for uncertainty and noise in the weak supervision annotation process. If the values in the covariance matrix Σ are too large, then the Gaussian weighing function would be too flat and it would fail to penalise bounding boxes that are far from $[cc_i^x, cc_i^y]$. In contrast, if the values are too small, then only the bounding boxes who centres are almost exactly the same as the centre supervision would be allowed. This would then result in suboptimal results when the centre supervision is noisy. The Σ is fixed to:

$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

before starting any experiments since it seems to be a set of reasonable values. The Gaussian function should penalise the same in any direction which is why a spherical covariance is used. We have not changed or manually tuned these values for any of the experiments, yet it turns out that this simple spherical covariance gives good results. Visualisation of this spatial penalty term is illustrated in Figures 6.5 and 6.6.

6.3.3 Training the Scene-specific Detector

At the end of the cue fusion and optimization discussed in Section 6.3.2, a set of bounding box annotations, $\mathbb{S}_{\text{bbox}}$, for the target video, $\mathbb{V}$, is obtained.

In order to get the scene-specific positive data $\mathbf{X}_t^+$, the patches corresponding to $\mathbb{S}_{\text{bbox}}$ are cropped out and feature extraction function is run on each of the patches (after appropriate resizing). These feature vectors make up $\mathbf{X}_t^+$.

The scene-specific negative data $\mathbf{X}_t^-$ are obtained in a similar fashion except that patches are cropped out from regions that do not intersect with $\mathbb{S}_{\text{bbox}}$.

A scene-specific pedestrian detector $\mathcal{C}_t$ can now be trained using $\mathbf{X}_t^+$ and $\mathbf{X}_t^-$. Any type of classifier can be used. We (again) use linear SVM for simplicity. Not only the classifier type but the feature extraction function is also general and it does not have to be the same as the one used during the unsupervised prior learning.

Figure 6.3: CUHK Square dataset: Random samples of patches corresponding to foreground objects which form $\mathbf{X}_t^{\text{fg}}$ (after feature extraction). Generalising (*i.e.* learning) over these examples serves as a *pedestrian prior* for the CUHK Square scene.

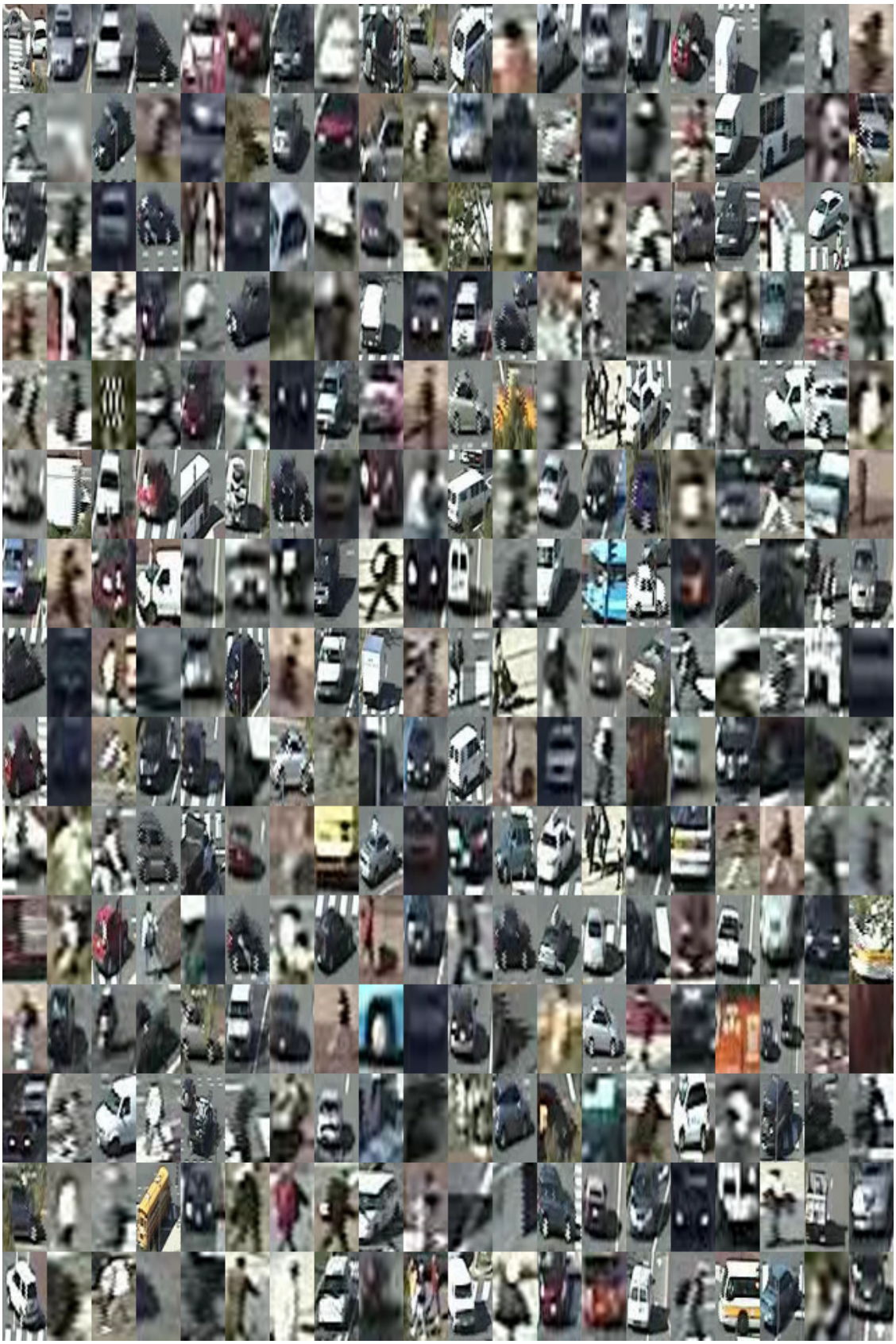


Figure 6.4: MIT Traffic dataset: Random samples of patches corresponding to foreground objects which form $\mathbf{X}_t^{\text{fg}}$. Generalising (*i.e.* learning) over these examples serves as a *pedestrian prior* for the MIT traffic scene.

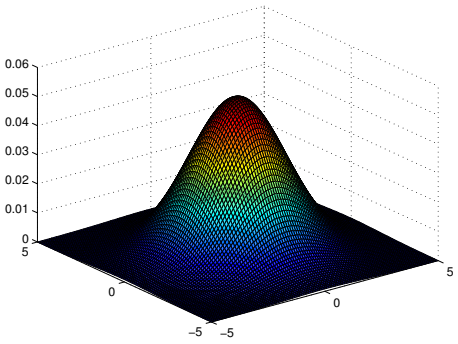
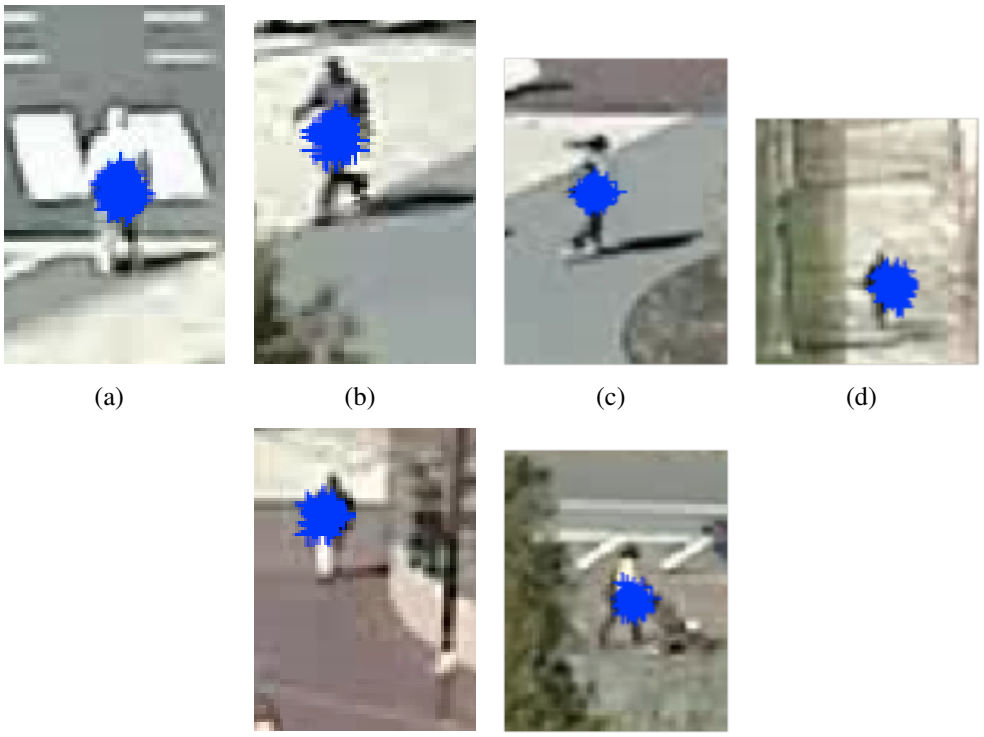


Figure 6.5: Bivariate normal distribution with mean $\mu = [0,0]$ and diagonal covariance matrix $\Sigma = \left[\begin{smallmatrix} 3 & 0 \\ 0 & 3 \end{smallmatrix}\right]$.



(a) (b) (c) (d)

(e) (f)

Figure 6.6: Visualisation of 100 samples from the bivariate normal distribution with mean μ at the weak supervision centre and diagonal covariance matrix $\Sigma = \left[\begin{smallmatrix} 3 & 0 \\ 0 & 3 \end{smallmatrix}\right]$ overlapped on some snippets of frames (a-f).

6.4 Experimental Results

We have used the challenging CUHK Square [110] and MIT Traffic [111] video datasets described in Chapters 4 and 5. However, this chapter, unlike Chapters 4 and 5, is tackling the problem of weakly supervised learning and *not* domain adaptation and hence there is no notion of "source dataset" and therefore the INRIA dataset is not involved in any of the experiments.

For each video dataset, we split it to two equal halves. During weakly-supervised training (including unsupervised prior learning), we only use the first half. The second half is kept purely for evaluating the resulting pedestrian detectors and the same protocol is used for evaluation as in Chapter 4 and Chapter 5, that is:

- 100 uniformly sampled frames annotated with groundtruth bounding boxes are used for evaluation.
- Recall-FPPI (False Positives Per Image) curves are used for performance comparison.
- PASCAL 50% overlap criterion [31] is used to decide whether the detected bounding boxes are correct with respect to the groundtruth bounding boxes.

We perform three different types of experiments on each dataset:

1. The pedestrian detector obtained by our weakly supervised algorithm.
2. The detector obtained by strong supervision (manual bounding box annotation).
3. The detector corresponding to the unsupervised prior (as described in Algorithm 6.2).

These experiments are respectively named *Weak supervision*, *Strong supervision* and *Unsupervised prior* in the curves shown in Figure 6.7. In addition, the cost comparison between the weak and strong supervisions is shown in Figure 6.8.

As illustrated, the detection performance of the proposed algorithm closely matches that of the strong supervision. Yet, the time it took to manually annotate training data for the proposed algorithm was less than one quarter of the time taken for the strong supervision.

This means that our algorithm reduces the manual human annotation effort by over 4 times to get the same performance as the standard strongly supervised training in literature. We also evaluated unsupervised prior in order to show the effectiveness of our

fusion and optimization framework. The unsupervised prior alone performs poorly; however, when fused with the weak supervision, the resulting detector has a much higher performance than the unsupervised prior.

The reason for the proposed algorithm performing very slightly lower than the strong supervision may be because of some minor inaccuracies in the generated bounding boxes (the output of the cue fusion and optimization), which result in very small bounding box misalignments (to the actual pedestrians) compared to strong supervision (where the bounding boxes are manually given by humans resulting in less such alignment errors).

There are a number of weaknesses associated with the proposed method in this chapter. Firstly, although the optimization algorithm automatically infers pedestrian scales as part of the optimization, there is still some implicit assumption that the range of these scales should not be too large. This can be improved by making the covariance of the spatial penalty term to depend on the (unknown) scale of the pedestrian. Secondly, the proposed may not work very well in highly crowded scenes where pedestrians occlude each other to a large degree. Moreover, it is important to note that although we have shown that the proposed weakly supervised method can reduce the manual annotation effort by over four times, a more rigorous study would be required to give a more reliable estimation of the expected cost saving.

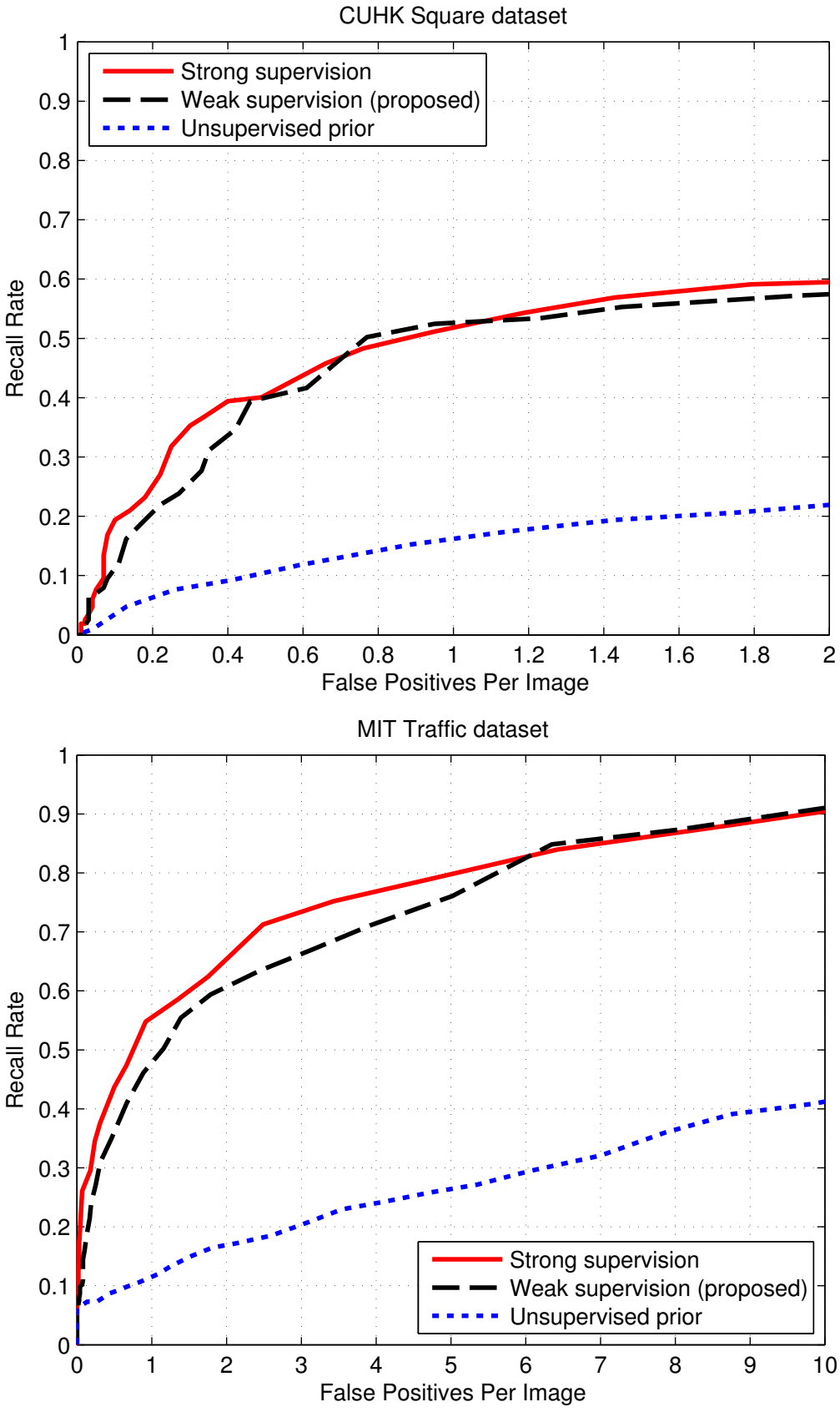


Figure 6.7: Detection performance curves for CUHK (left) and MIT (right)

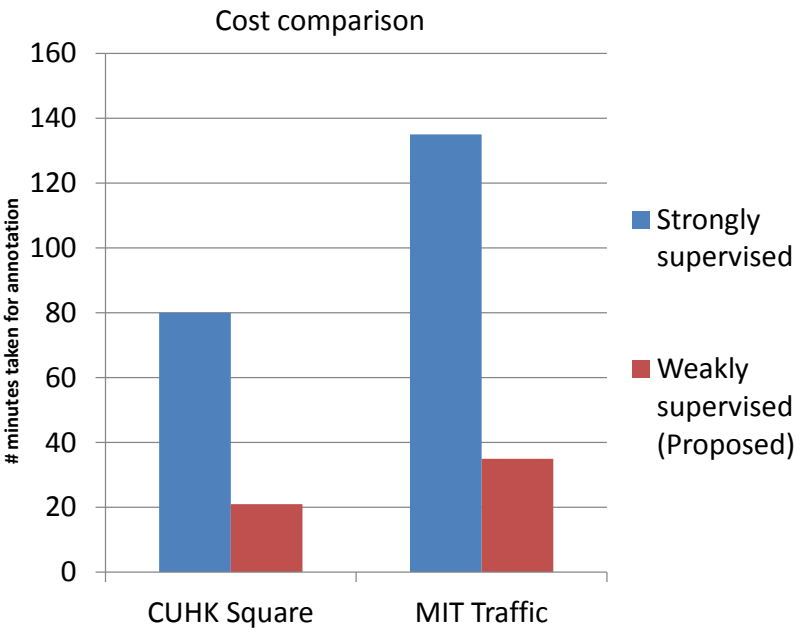


Figure 6.8: Cost comparison

6.5 Conclusion

We have proposed a novel weakly supervised learning algorithm for training pedestrian detectors for videos. The algorithm consists of learning an unsupervised prior using unlabelled data in the video and then fusing the prior with the weak supervision in an optimization framework to generate bounding box annotations. We have shown that the weakly supervised algorithm can reduce the amount of human annotation effort by over four times without sacrificing the accuracy of the resulting detector.

Chapter 7

Conclusion and Future Work

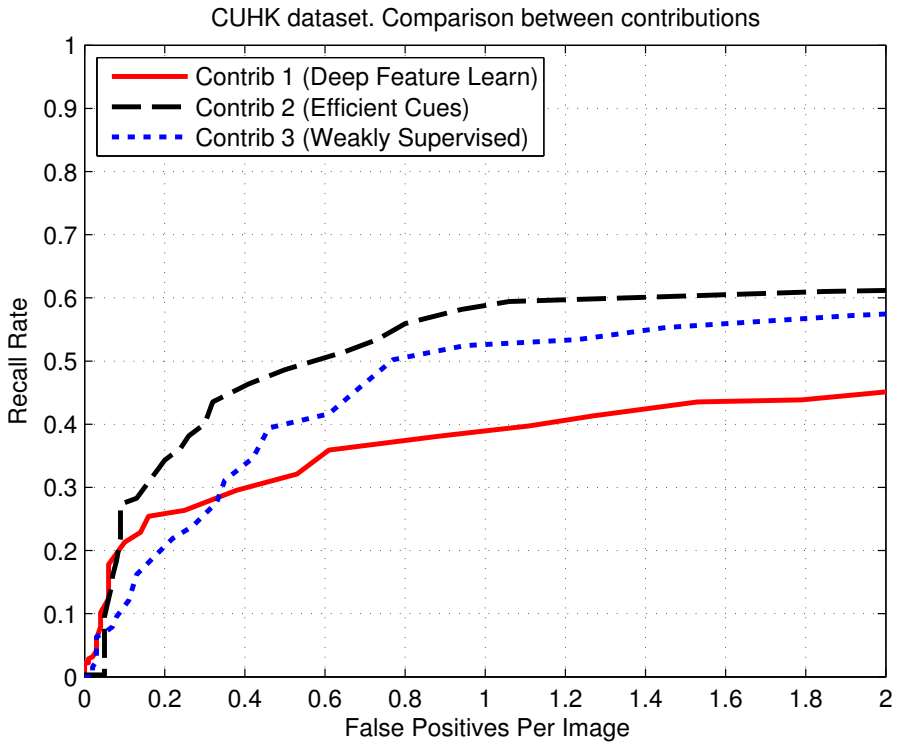
7.1 Overall Discussion

In Chapters 4, 5 and 6, we have proposed three different algorithms with the goal of minimising human effort in obtaining scene-specific pedestrian detectors. In each of these chapters, we have shown the effectiveness of the proposed algorithms by comparing with various relevant baselines and state-of-the-art methods. We now compare the contributions across these chapters and discuss the connections and relative performances among them.

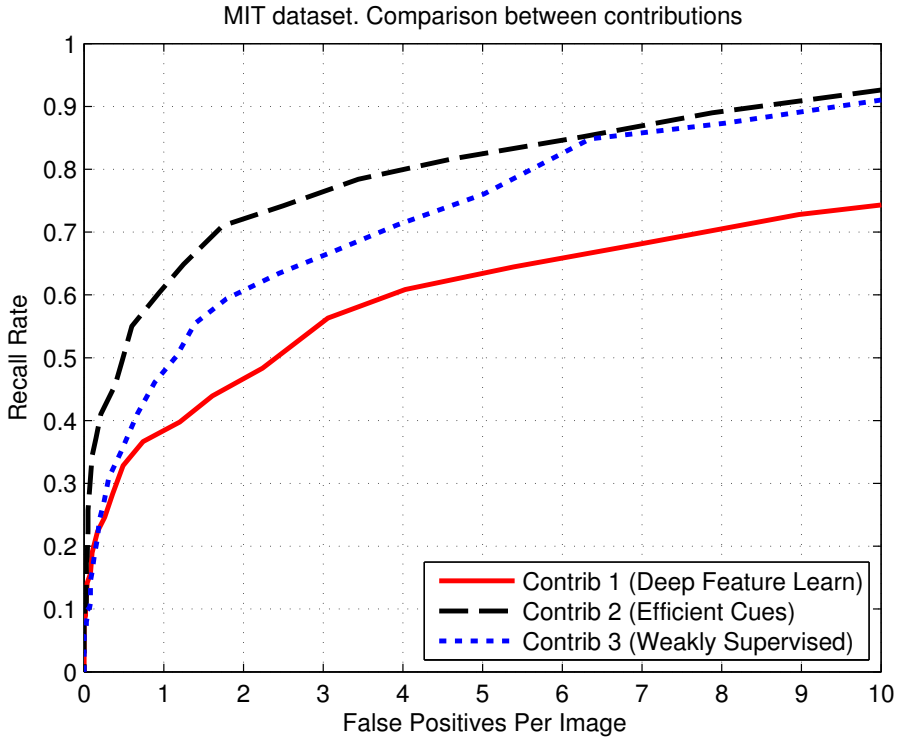
Figure 7.1 shows the comparisons. From the performance curves, the following observations could be made.

Comparing between the two domain adaptation methods (*i.e.* `Contrib 1` and `Contrib 2`), `Contrib 2`, which is the simple and efficient algorithm that makes effective use of cues in videos, performs much better than the one (*i.e.* `Contrib 1`) that explicitly models and learns the manifold.

The unsupervised domain adaptation method `Contrib 2` also outperforms the weakly supervised learning approach (`Contrib 3`) in both of the datasets. This shows that the domain adaptation algorithm `Contrib 2` is highly effective considering that it does *not* require any type of supervision (either weak or strong) in the target scene.



(a)



(b)

Figure 7.1: Comparison of the main novel contributions in the thesis. (a) is for the CUHK Square dataset and (b) is for the MIT Traffic dataset. `Contrib 1` corresponds to the contribution in the Chapter 4 of the thesis, `Contrib 2` is the contribution in Chapter 5 and `Contrib 3` is for Chapter 6.

The weakly supervised learning approach `Contrib 3` however outperforms `Contrib 1` on both datasets. Overall, the weakly supervised learning algorithm (`Contrib 3`) performs quite well in both of the datasets. This is useful in situations where a generic dataset (or detector) is not available.

7.2 Summary of Strengths and Weaknesses, and Recommendations

The summary of the strengths and the weaknesses of the methods presented in Chapters 4, 5 and 6 is shown in Table 7.1. Based on these strengths and weaknesses, we now make some recommendations on the best way to choose the proposed methods in practical situations.

If the target dataset (*i.e.* domain) is a video captured with a static camera, it is best to use the non-iterative self-training algorithm proposed in Chapter 5 because it makes maximum use of cues available in video, resulting in the highest domain adaptation accuracy. Moreover, not only it is efficient and fast during training (*i.e.* during domain adaptation), it is also very fast at test time since there is no need to perform expensive feature projection (as required by the domain adaptation algorithm using deep learning in Chapter 4). In addition to this, if there is a generic detector but if the corresponding generic dataset is not available, the non-iterative self-training method (proposed in Chapter 5) should be used since the algorithm in Chapter 4 requires the generic dataset to be present.

If smooth spatio-temporal constraints cannot be reliably exploited in the target domain (either due to the video camera recording at very low frame rates, due to the presence of sufficiently large camera movements or due to the fact that the target domain is a set of static image collections with no temporal connections), we would recommend using proposed domain adaptation using deep learning. However, with this approach, if faster pedestrian detection is desired, we would recommend that during test time, rather than exhaustive sliding window detection, some other methods (such as 3D or ground plane information) should be used to limit the the number of sliding windows that need to be evaluated.

Finally, in a situation where neither the generic dataset nor the generic detector is present, the weakly-supervised learning approach presented in Chapter 6 should be used.

Chap	Strengths	Weaknesses
4	• Does not require background subtraction or tracking. So it is not affected by the errors in these processes. • Can work for static images as the target domain; does not require it to be a (continuous) video.	• Requires feature projection at test time. This can be expensive for a the sliding window detector where the feature projection and the classifier need to be evaluated on hundreds of thousands or even millions of sliding windows for each frame in the video. • Slow domain adaptation (i.e. slow training); most of the time is taken by feature/manifold learning. Although the speed can be improved by implementing the deep learning on GPUs, this is not trivial and this does not work across all types of GPUs. • The resulting scene-specific detector attempts to do well on both the generic dataset and the target dataset. Intuitively, better domain adaptation performance could be achieved by focussing to do well only on the target dataset.
5	• Highest domain adaptation performance (i.e. higher than the algorithm in Chapter 4 and state-of-the-art works). • Does not require feature projection at test time. Therefore, very fast at test time. • Fast and efficient during domain adaptation. Can be parallelized non-trivially. • Does not require the generic dataset during adaptation; just the generic detector is sufficient. This is advantageous in situations where the generic dataset is large and therefore costly to transmit to various sites for domain adaptation.	• May not work well for very crowded scenes such as in densely-populated shopping malls due to too large a number of background proposals coming from wrongly connected foreground blobs and short-term tracking errors. • May be sensitive to large camera movements.
6	• Does not require the generic dataset or the generic detector to generate a scene-specific pedestrian detector. Only requires the target scene. This is advantageous in situations where the generic dataset/detector is not available for some reason.	• Requires (weak) supervision in the target scene in contrast to the domain adaptation approaches (in Chapters 4 and 5) that do not require any supervision in the target scene. • Although the optimization algorithm automatically infers pedestrian scales as part of the optimization, there is still some implicit assumption that the range of these scales should not be too large.

Table 7.1: Summary of the strengths and weaknesses of the proposed methods in Chapters 4, 5 and 6.

7.3 Conclusion

In this thesis, three main novel algorithms have been proposed to generate scene-specific pedestrian detectors for video scenes. These three contributions encompass two machine learning paradigms: domain adaptation and weakly supervised learning.

The first and second contributions are unsupervised domain adaptation methods which tune a generic pedestrian detector trained on a (high-resolution) image dataset to specific (low-resolution) video scenes. These two domain adaptation algorithms utilise the knowledge available in a generic dataset and do not require any labelled data in the target scene. We showed that the proposed algorithms outperform baselines and state-of-the-art algorithms.

The third contribution is a weakly supervised learning algorithm that requires supervision in the target dataset albeit a "weaker" and easier form of supervision, namely approximate centre locations of objects. This method is beneficial when a generic dataset is not readily available. Despite requiring much less supervision labour, the method performs almost on par with traditional strong supervision in the form of precise bounding box annotation.

All the proposed algorithms have been evaluated on two very challenging video datasets that consist of multiple classes of small objects including pedestrians and vehicles. We have also compared the proposed algorithms with each other and discussed the results.

7.4 Summarised Answers to Research Questions

In this section, we set out to summarise the answers to the research questions that were initially posed in Section 1.2.

We have shown that detector adaptation for pedestrian detection can be performed by using only the manifold assumption of data. Using the algorithm proposed in Chapter 4, deep learning can be effectively used for detector adaptation for videos. Our proposed algorithm outperforms a baseline and a state-of-the-art algorithm that make use of a similar manifold assumption.

In Chapter 5, we have shown that by using spatio-temporal cues and making effective use of information available from videos, we can simplify the job of domain adaptation and have a simple and efficient system that gives superior performance over pure manifold learning approaches that do not use such cues.

We have shown in Chapter 5 that if a generic dataset is not available and domain adaptation cannot be performed and therefore supervision has to be given in the target

scene, one of the faster ways is to annotate only approximate centres of objects rather than the standard bounding boxes. We have shown that for videos, we can learn useful priors from the scene and use them in an efficient optimization framework to convert the given weak supervision to strong supervision. This results in detectors that performs nearly as well as traditional strong supervision, yet the proposed weak supervision is over four times faster than traditional supervision.

7.5 Limitations

There are a few limitations associated with the proposed methods.

- The algorithms proposed in Chapters 5 and 6 only work with videos recorded using static cameras. Although this covers a very large range of scenarios in real life, it would be interesting to explore domain adaptation methods for scenes requiring dynamic cameras such as networks of moving surveillance cameras.

- For biased sampling algorithm in Chapter 4, there may be too much bias towards the false positives of the detector and this may be undesirable. The sampling process would accordingly miss other types of patches that the detector can correctly reject as non-pedestrian.

- In Chapter 4, the layer size of the deep network architecture decreases by a factor of two until the bottleneck layer is reached. Although this is a reasonable assumption, there is no evidence that this rate of decrease is optimal.

- In Chapter 6, the covariance of the bivariate normal distribution used to penalise sliding windows that are far from the weak supervision centres is fixed regardless of the scale of the windows. However, it may make more sense to vary the covariance depending on the scale of the windows and doing so might give better results.

7.6 Future Work

In the future, it would be interesting to extend the current work in several directions as outlined below.

Firstly, it would be interesting to investigate the effects of domain adaptation for different views of the same scene. For example, for a traffic intersection with videos recorded from different cameras that are capturing different views and angles of the same place, it would be interesting to characterise how much domain adaptation is required between these different recordings, as compared to totally different scenes and places.

Currently, our weakly supervised learning makes use of an unsupervised prior that gives some partial information about pedestrians in the scene. It would interesting to replace this unsupervised prior with a small amount of labelled data and compare the results. Moreover, by doing this, we can lift the restriction about requiring the camera to be static. Furthermore, gradually reducing supervision for a machine learning task is interesting in its own right: firstly start with a very small amount of very strong supervision, then use that as a prior to help learn with a larger amount of moderately strong supervision, then use that as a prior for gradually weaker and weaker forms of supervision with larger and larger amounts of data.

It would be of benefit to test our current domain adaptation algorithms using different lengths of video during the adaptation stage and to see how the results change with differing availability of video lengths. Moreover, it may be of interest to characterise the minimal amount of offline video required for a successful domain adaptation.

For the algorithm proposed in Chapter 5, instead of using background subtraction and Connected Component Analysis for bounding box proposal generation, there is a possibility to develop more sophisticated methods based on joint spatio-temporal segmentation.

In learning an unsupervised prior for weakly supervised training of a pedestrian detector in Chapter 6, it would be interesting to investigate the effects of using various non-linear classifiers instead of a linear classifier. It would be interesting to see if this would give a better unsupervised prior and if so, to observe how much improvement in the final detector this better prior would make. Moreover, outlier detection may also be useful for the unsupervised prior learning to filter out "bad" foreground objects.

Finally, it would be interesting to see the results of applying the algorithms proposed in this thesis to object categories other than pedestrians.

7.7 Publications

The following papers have been published in refereed conference proceedings:

1. Kyaw Kyaw Htike and David Hogg, "Efficient non-iterative domain adaptation of pedestrian detectors to video scenes", *International Conference on Pattern Recognition*, ICPR, 2014.

2. Kyaw Kyaw Htike and David Hogg, "Weakly supervised pedestrian detector training by unsupervised prior learning and cue fusion in videos", *International Conference on Image Processing*, ICIP, 2014.

3. Kyaw Kyaw Htike and David Hogg, "Unsupervised detector adaptation by joint dataset feature learning", *International Conference on Computer Vision and Graphics*, ICCVG, 2014.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012.

[2] Karim Ali, David Hasler, and François Fleuret. Flowboost - appearance learning from sparsely annotated video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1433–1440, 2011.

[3] Alexander Andreopoulos and John K Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, pages 827–891, 2013.

[4] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support Vector Machines for Multiple-instance Learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 561–568, 2002.

[5] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2903–2910. IEEE, 2012.

[6] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[7] Matthew Blaschko, Andrea Vedaldi, and Andrew Zisserman. Simultaneous object detection and ranking with weak supervision. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 235–243, 2010.

[8] John Blitzer, Ryan T. McDonald, and Fernando Pereira. Domain adaptation with Structural Correspondence Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 120–128, 2006.

[9] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *The Annual Conference on Computational Learning Theory (COLT)*, pages 92–100. ACM, 1998.

[10] Biswajit Bose and W. Eric L. Grimson. Improving object classification in far-field video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 181–188, 2004.

[11] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 161–168, 2007.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Sebastian Brutzer, Benjamin Höferlin, and Gunther Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1937–1944, 2011.

[14] Nicholas J. Butko and Javier R. Movellan. Optimal scanning for faster object detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2751–2758, 2009.

[15] João Carreira and Cristian Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[16] Yixin Chen and James Z Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5:913–939, 2004.

[17] Sen-Ching S Cheung and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In *Proceedings of SPIE*, volume 5308, pages 881–892, 2004.

[18] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[19] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[22] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–466, 2010.

[23] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[24] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 645–659, 2012.

[25] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[26] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2009.

[27] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[28] Ian Endres and Derek Hoiem. Category independent object proposals. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 575–588, 2010.

[29] Markus Enzweiler and Dariu M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009.

[30] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[31] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[32] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR : A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[33] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[34] Michael C Ferris and Todd S Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2002.

[35] James Ferryman and Ali Shahrokni. PETS 2009: Dataset and challenge. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, pages 1–6, Dec 2009.

[36] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[37] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of Boosting. *Annals of Statistics*, 28:337–407, 1998.

[38] Carolina Galleguillos, Boris Babenko, Andrew Rabinovich, and Serge J. Belongie. Weakly supervised object localization with stable segmentations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 193–207, 2008.

[39] Dariu M. Gavrila and Stefan Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59, 2007.

[40] Paul Geladi and Bruce R Kowalski. Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

[41] David Geronimo, Antonio M Lopez, Angel Domingo Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1239–1258, 2010.

[42] Ross B Girshick, Pedro F Felzenszwalb, and David A Mcallester. Object detection with grammar models. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 442–450, 2011.

[43] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[44] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073, 2012.

[45] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 999–1006, 2011.

[46] Helmut Grabner, Peter M. Roth, and Horst Bischof. Is pedestrian detection really a hard task? In *Proc. IEEE Intern. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pages 1–8, 2007.

[47] Giovanni Gualdi, Andrea Prati, and Rita Cucchiara. Multi-stage sampling with boosting cascades for pedestrian detection in images and videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 196–209, 2010.

[48] Giovanni Gualdi, Andrea Prati, and Rita Cucchiara. A multi-stage pedestrian detection using monolithic classifiers. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 267–272, 2011.

[49] Marko Heikkilä and Matti Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):657–662, 2006.

[50] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[51] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[52] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[53] Rebecca Hwa. Supervised grammar induction using training data with limited constituent information. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 73–79, 1999.

[54] Vidit Jain and Sachin Sudhakar Farfade. Adapting classification cascades to new domains. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 105–112, 2013.

[55] Omar Javed, Saad Ali, and Mubarak Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 696–701, 2005.

[56] Jing Jiang. A literature survey on domain adaptation of statistical classifiers. `http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/da_survey.html`, March 2008. [Online; last accessed 1-January-2014].

[57] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226. ACM, 2006.

[58] Aniruddha Kembhavi, Behjat Siddiquie, Roland Miezianko, Scott McCloskey, and Larry S. Davis. Incremental multiple kernel learning for object recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 638–645, 2009.

[59] Li Fei-Fei Daphne Koller Kevin Tang, Vignesh Ramanathan. Shifting weights: Adapting object detectors from image to video. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 638–646, 2012.

[60] Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1785–1792, 2011.

[61] Steve Lawrence, C. Lee Giles, and Ah Chung Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical report, 1996.

[62] Quoc Le, Tamas Sarlos, and Alex Smola. Fastfood–approximating kernel expansions in loglinear time. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 244–252, 2013.

[63] Kuang-Chih Lee, Jeffrey Ho, and David Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

[64] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc J. Van Gool. Dynamic 3D scene analysis from a moving vehicle. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[65] Anat Levin, Paul A. Viola, and Yoav Freund. Unsupervised improvement of visual detectors using co-training. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 626–633, 2003.

[66] Elizaveta Levina and Peter J Bickel. Maximum likelihood estimation of intrinsic dimension. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 777–784, 2004.

[67] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[68] Dong C. Liu, Jorge Nocedal, and Dong C. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[69] Tomasz Malisiewicz and Alexei A. Efros. Improving spatial support for objects via multiple segmentations. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10, 2007.

[70] James Martens. Deep learning via Hessian-free optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 735–742, 2010.

[71] David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 337–344. Association for Computational Linguistics, 2006.

[72] John Willard Milnor. *Characteristic classes*. Princeton University Press, 1974.

[73] Fatemeh Mirrashed, Vlad I. Morariu, and Larry S. Davis. Sampling for unsupervised domain adaptive object detection. In *IEEE International Conference on Image Processing (ICIP)*, pages 3288–3292, 2013.

[74] Fatemeh Mirrashed and Mohammad Rastegari. Domain adaptive classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2608–2615, 2013.

[75] Vinod Nair and James J. Clark. An unsupervised, online learning framework for moving object detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 317–324, 2004.

[76] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Andrew Ng, and Quoc V Le. On optimization methods for deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 265–272, 2011.

[77] Minh Hoai Nguyen, Lorenzo Torresani, Fernando de la Torre, and Carsten Rother. Weakly supervised discriminative localization and classification: a joint learning process. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1925–1932, 2009.

[78] Mustafa Özuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.

[79] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1187–1192, 2009.

[80] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[81] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1307–1314. IEEE, 2011.

[82] Marco Pedersoli, Andrea Vedaldi, and Jordi Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1353–1360. IEEE, 2011.

[83] Massimo Piccardi. Background subtraction techniques: a review. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3099–3104, 2004.

[84] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3282–3289, 2012.

[85] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, volume 3, pages 1177–1184, 2007.

[86] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1313–1320, 2008.

[87] Esa Rahtu, Juho Kannala, and Matthew B. Blaschko. Learning a category independent object detection cascade. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1052–1059, 2011.

[88] Brian Roark and Michiel Bacchiani. Supervised and unsupervised PCFG adaptation to novel domains. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*, pages 126–133, 2003.

[89] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 29–36, 2005.

[90] Peter M Roth, Helmut Grabner, Danijel Skocaj, H Bischol, and Aleš Leonardis. On-line conservative learning for person detection. In *Proc. IEEE Intern. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pages 223–230, 2005.

[91] Peter M. Roth, Sabine Sternig, Helmut Grabner, and Horst Bischof. Classifier grids for robust adaptive object detection. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2727–2734, 2009.

[92] Todd Rowland. Manifold. `http://mathworld.wolfram.com/Manifold.html`. [Online; last accessed 1-January-2014].

[93] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA, 1988.

[94] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–226, 2010.

[95] Hanan Samet and Markku Tamminen. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):579–586, July 1988.

[96] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 343–351, 2013.

[97] Mark Schmidt. minFunc: unconstrained optimization of differentiable real-valued multivariate functions. `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`. [Online; last accessed 1-January-2013].

[98] Mark Segal. Machine learning benchmarks and random forest regression. *Center for Bioinformatics & Molecular Biostatistics*, 2004.

[99] Pramod Sharma, Chang Huang, and Ram Nevatia. Unsupervised incremental learning for improved object detection in a video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3298–3305, 2012.

[100] Pramod Sharma and Ram Nevatia. Efficient detector adaptation for object detection in a video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3261, 2013.

[101] Guang Shu, Afshin Dehghan, and Mubarak Shah. Improving an object detector and extracting regions using superpixels. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3721–3727, 2013.

[102] S. Stalder, H. Grabner, and LV Gool. Exploring context to learn scene specific object detectors. In *Proc. IEEE Intern. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pages 63–70, 2009.

[103] Alexander Statnikov, Lily Wang, and Constantin Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9(1):319, 2008.

[104] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[105] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528, 2011.

[106] Jasper R. R. Uijlings, Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[107] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, and Arnold W. M. Smeulders. Segmentation as selective search for object recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1879–1886, 2011.

[108] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–492, 2012.

[109] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1–8, 2013.

[110] Meng Wang, Wei Li, and Xiaogang Wang. Transferring a generic pedestrian detector towards specific scenes. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3274–3281, 2012.

[111] Meng Wang and Xiaogang Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3401–3408, 2011.

[112] Xiaogang Wang, Xiaoxu Ma, and W. Eric L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):539–555, 2009.

[113] Xiaoyu Wang, Gang Hua, and Tony X. Han. Detection by detections: Non-parametric detector adaptation for a video. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 350–357, 2012.

[114] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 18–32, 2000.

[115] Bo Wu and Ram Nevatia. Improving part based object detection by unsupervised, online boosting. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[116] Jian Yao and Jean-Marc Odobez. Multi-layer background subtraction based on color and texture. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[117] Liang Zhao and Charles E. Thorpe. Stereo- and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3):148–154, 2000.